SYBASE®

ユーザーズ・ガイド

Sybase ETL

4.8

ドキュメント ID：DC00906-01-0480-01

改訂：2009 年 2 月

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

目次

はじめに

対象読者

このマニュアルは、Sybase ETL Development のユーザを対象にしています。

このマニュアルの使用方法

このマニュアルには、以下の章があります。

- 「第 1 章 Sybase ETL」では、Sybase ETL アーキテクチャと Sybase ETL Development および Sybase ETL サーバの機能セットの概要について説明します。
- 「第 2 章 使用開始にあたって」では、Sybase ETL の使用を開始する方法について説明します。Sybase ETL Development インタフェースの基本について説明し、このインタフェースを使用して実行できる機能についても説明します。
- 「第 3 章 プロジェクトとジョブ」では、プロジェクトおよびジョブの作成、シミュレーション、実行について説明します。シミュレーション・モードの使用方法と、テンプレートを使用したプロジェクトおよびジョブの作成方法について説明します。
- 「第 4 章 高度な概念とツール」では、設計作業を簡単にする組み込みツールについて説明します。
- 「第 5 章 コンポーネント」では、プロジェクトおよびジョブの作成に使用する Sybase ETL コンポーネントについて説明します。
- 「第 6 章 Sybase ETL サーバ」では、Sybase ETL サーバの使用方法について説明します。
- 「付録 A 関数リファレンス」では、Sybase ETL で使用できる組み込み関数について説明します。
- 「付録 B 接続パラメータ」では、データベース設定オプションについて説明します。また、一部のサポートされているインタフェースに関する追加情報も提供します。
- 「付録 C 緩やかに変化する次元に対する ETL の使用」では、いくつかの一般的な SCD シナリオなど、緩やかに変化する次元 (SCD) について説明し、Sybase ETL を使用してこれらのシナリオを実装する方法についても説明します。

- 「付録 D ベスト・プラクティス」では、Sybase ETL を操作するための推奨事項とガイドラインについて説明します。

関連マニュアル

詳細については、次のマニュアルを参照してください。

- 『Sybase ETL 新機能ガイド』– Sybase ETL 4.8 の新しい機能について説明しています。
- 『Sybase ETL リリース・ノート』– マニュアルには記載できなかった最新の情報が記載されています。

その他の情報

Sybase Getting Started CD、SyBooks™ CD、Sybase Product Manuals Web サイトを利用すると、製品について詳しく知ることができます。

- Getting Started CD には、PDF 形式のリリース・ノートとインストール・ガイド、SyBooks CD に含まれていないその他のマニュアルや更新情報が収録されています。この CD は製品のソフトウェアに同梱されています。Getting Started CD に収録されているマニュアルを参照または印刷するには、Adobe Acrobat Reader が必要です (CD 内のリンクを使用して Adobe の Web サイトから無料でダウンロードできます)。
- SyBooks CD には製品マニュアルが収録されています。この CD は製品のソフトウェアに同梱されています。Eclipse ベースの SyBooks ブラウザを使用すれば、使いやすい HTML 形式のマニュアルにアクセスできます。

  一部のマニュアルは PDF 形式で提供されています。それらのマニュアルは SyBooks CD の PDF ディレクトリに収録されています。PDF ファイルを開いたり印刷したりするには、Adobe Acrobat Reader が必要です。

  SyBooks のインストールと起動の方法については、Getting Started CD の『SyBooks インストール・ガイド』、または SyBooks CD の *README.txt* ファイルを参照してください。
- Sybase Product Manuals Web サイトは、SyBooks CD のオンライン版であり、標準の Web ブラウザを使ってアクセスできます。また、製品マニュアルのほか、EBFs/Maintenance、Technical Documents、Case Management、Solved Cases、ニュース・グループ、Sybase Developer Network へのリンクもあります。

  Sybase Product Manuals Web サイトにアクセスするには、Product Manuals (http://www.sybase.com/support/manuals/) にアクセスしてください。

Web 上の Sybase 製品の動作確認情報

Sybase Web サイトの技術的な資料は頻繁に更新されます。

❖ **製品動作確認の最新情報にアクセスする**

1 Web ブラウザで Technical Documents (http://www.sybase.com/support/techdocs/) を指定します。

2 [Certification Report] をクリックします。

3 [Certification Report] フィルタで製品、プラットフォーム、時間枠を指定して [Go] をクリックします。

4 [Certification Report] のタイトルをクリックして、レポートを表示します。

❖ **コンポーネント動作確認の最新情報にアクセスする**

1 Web ブラウザで Availability and Certification Reports (http://certification.sybase.com/) を指定します。

2 [Search By Base Product] で製品ファミリと製品を選択するか、[Search by Platform] でプラットフォームと製品を選択します。

3 [Search] をクリックして、入手状況と動作確認レポートを表示します。

❖ **Sybase Web サイト (サポート・ページを含む) の自分専用のビューを作成する**

MySybase プロファイルを設定します。MySybase は無料サービスです。このサービスを使用すると、Sybase Web ページの表示方法を自分専用にカスタマイズできます。

1 Web ブラウザで Technical Documents (http://www.sybase.com/support/techdocs/) を指定します。

2 [MySybase] をクリックし、MySybase プロファイルを作成します。

Sybase EBF とソフトウェア・メンテナンス

❖ **EBF とソフトウェア・メンテナンスの最新情報にアクセスする**

1 Web ブラウザで Sybase Support Page (http://www.sybase.com/support) を指定します。

2 [EBFs/Maintenance] を選択します。ユーザ名とパスワードの入力が求められたら、MySybase のユーザ名とパスワードを入力します。

3 製品を選択します。

4 時間枠を指定して [Go] をクリックします。EBF/Maintenance リリースの一覧が表示されます。

鍵のアイコンは、「Technical Support Contact」として登録されていないため、一部の EBF/Maintenance リリースをダウンロードする権限がないことを示しています。未登録ではあるが、Sybase 担当者またはサポート・コンタクトから有効な情報を得ている場合は、[Edit Roles] をクリックして、「Technical Support Contact」役割を MySybase プロファイルに追加します。

5 EBF/Maintenance レポートを表示するには [Info] アイコンをクリックします。ソフトウェアをダウンロードするには製品の説明をクリックします。

表記規則

このマニュアルで使用する構文の表記規則は、次のとおりです。

キー	定義
コマンドおよび**メソッド**	コマンド名、コマンドのオプション名、ユーティリティ名、ユーティリティのフラグ、Java メソッド／クラス／パッケージ、その他のキーワードは、小文字の Arial フォントで表記する。
変数	斜体は次の内容を示す。 • *myServer* などのプログラム変数 • 次のような置換の必要がある入力テキストの一部 *Server*.log • ファイル名
[ファイル] - [保存]	メニュー名とメニュー項目はプレーン・テキストで表記される。ハイフンはメニューの選択肢間の移動を示す。たとえば、[ファイル] - [保存] は、[ファイル] メニューから [保存] を選択することを示す。
`package 1`	等幅フォントは次の内容を示す。 • GUI インタフェースやコマンド・ラインに入力する情報、またはプログラムのテキストとして入力する情報 • サンプル・プログラムのフラグメント • サンプル出力のフラグメント

アクセシビリティ機能

このマニュアルには、アクセシビリティを重視した HTML 版もあります。この HTML 版マニュアルは、スクリーン・リーダーで読み上げる、または画面を拡大表示するなどの方法により、その内容を理解できるよう配慮されています。

Sybase ETL の HTML マニュアルは、連邦リハビリテーション法第 508 条のアクセシビリティ規定に準拠していることがテストにより確認されています。第 508 条に準拠しているマニュアルは通常、World Wide Web Consortium (W3C) の Web サイト用ガイドラインなど、米国以外のアクセシビリティ・ガイドラインにも準拠しています。

注意 アクセシビリティ・ツールを効率的に使用するには、設定が必要な場合があります。一部のスクリーン・リーダーは、テキストの大文字と小文字を区別して発音します。たとえば、すべて大文字のテキスト (ALL UPPERCASE TEXT など) はイニシャルで発音し、大文字と小文字の混在したテキスト (MixedCase Text など) は単語として発音します。構文規則を発音するようにツールを設定することをおすすめします。詳細については、ツールのマニュアルを参照してください。

Sybase のアクセシビリティに対する取り組みについては、Sybase Accessibility (http://www.sybase.com/accessibility) を参照してください。Sybase Accessibility サイトには、第 508 条と W3C 標準に関する情報のリンクもあります。

不明な点があるときは

Sybase ソフトウェアのインストール環境ごとに、Sybase 製品の保守契約を結んでいるサポート・センタとの連絡担当者がいます。マニュアルやオンライン・ヘルプで解決できない問題がある場合は、この担当者を通して最寄りの Sybase のサポート・センタまでご連絡ください。

第 1 章

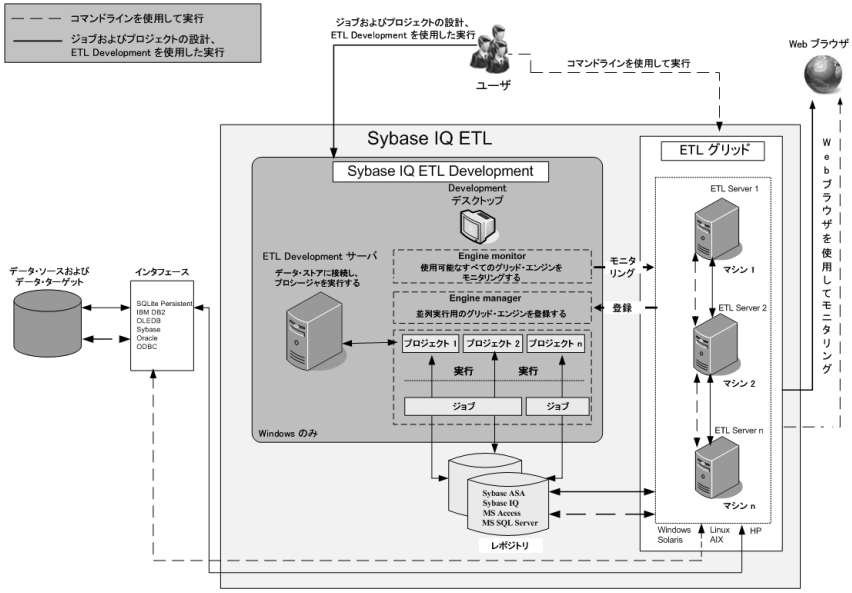
Sybase ETL

Sybase ETL では、広範な変換機能を使用して、複数の異機種データ・ソースからデータを抽出し、そのデータを 1 つまたは複数のデータ・ターゲットにロードできます。そのスケーラブルなグリッド・アーキテクチャによって、さまざまなオペレーティング・システムやコンピュータにまたがる並列的な変換処理が可能になります。

Sybase ETL には次の機能があります。

- データ抽出 – さまざまなデータ・ソースからデータを抽出します。
- データ変換 – データ・ストリームの変換、消去、結合、分割を行います。
- データ・ロード – update、insert、delete、または bulk copy 文を使用して、データをターゲット・データベースにロードします。

Sybase ETL アーキテクチャ



Sybase ETL のコンポーネント

Sybase ETL には、Sybase ETL Development と Sybase ETL サーバが含まれます。

Sybase ETL Development (Windows でのみ使用可能) は、データ変換プロジェクトおよびジョブの作成および設計を目的としたグラフィカル・ユーザ・インタフェース (GUI) ツールです。このツールには、ETL 変換フローの開発期間を短縮するために設計された完全なシミュレーション環境とデバッグ環境が備えられています。

Sybase ETL サーバは、スケーラブルな分散型のグリッド・エンジンです。データ・ソースに接続し、Sybase ETL Development で設計された変換フローを使用して、データ・ターゲットへのデータの抽出およびロードを行います。詳細については、「Sybase ETL サーバ」(225 ページ) を参照してください。

Sybase ETL Development には、データベースへの接続やプロシージャの実行など、実際の処理を制御する ETL Development サーバが含まれています。ジョブとプロジェクトの並列実行を行うために、ご使用のネットワーク内の異なるオペレーティング・システムに複数の ETL サーバを追加できます。各サーバは、他のすべてのピア・サーバに対して特定のサービスを公開します。Sybase ETL では、グリッド上のさまざまなサーバを使用してプロジェクトおよびジョブを並列的に実行するので、変換速度のスケーラビリティが向上します。

Sybase ETL サーバは、インタフェースと呼ばれるメソッドまたはドライバを使用して、送信先または送信元データベースに接続します。サポートされているインタフェースの他に、Sybase IQ への接続に使用される ODBC ドライバが Sybase IQ インストーラによって自動的にインストールされます。サポートされるその他のインタフェースをインストールするには、それぞれのベンダのマニュアルを参照してください。

注意 Sybase ETL Development を使用して開発されたプロジェクトとジョブの並列実行を行うには、独立した実行プログラムとして使用可能な Sybase ETL サーバをインストールしてください。

コマンド・ラインを使用して、サポートされているすべてのプラットフォームでジョブとプロジェクトを実行できます。詳細については、「ETL サーバを使用したプロジェクトおよびジョブの実行」(229 ページ) を参照してください。

ETL サーバの登録

グリッドに追加するすべての ETL サーバを登録する必要があります。Sybase ETL Development ツール内で使用できる Engine Manager を使用して、サーバを登録できます。詳細については、「複数のエンジンの使用によるジョブ実行時間の短縮」(85 ページ) を参照してください。

ETL サーバのモニタリング

Sybase ETL Development ツール内で使用できる Engine Monitor を使用して、サーバをお使いのネットワークでモニタリングできます。詳細については、「Engine Monitor」(88 ページ) を参照してください。コマンド・ライン起動で Web ブラウザを使用すると、リモート・プロジェクトとリモート・ジョブをモニタリングできます。詳細については、「Web ブラウザを使用したプロジェクトとジョブのモニタリング」(232 ページ) を参照してください。

注意 このマニュアルでは、グリッド・エンジンと ETL サーバという用語は同じ意味で使用されています。

Sybase ETL 概念

この項では、Sybase ETL の概念について説明します。

プロジェクトとジョブ

プロジェクトとは、コンポーネント、リンク、変換規則の集合です。各プロジェクトには、プロジェクトの実行時に順番にシミュレートまたは実行される 1 つまたは複数の手順が含まれています。シミュレートまたは実行時に、コンポーネントはさまざまなデータ・ソースに接続し、そのデータ・ソースから変換規則に基づいて変換対象のデータを読み込みます。プロジェクトは、自由に配置できる各種のコンポーネントで構成されます。コンポーネントは、[Component Store] のセクションから [Design] ウィンドウにドラッグすることによってプロジェクトに追加できます。

複数のプロジェクトをジョブ内で順番に実行するか並列に実行できます。ジョブはプロジェクトの実行順序を制御します。ジョブはスケジュールとモニタが可能です。

プロジェクトとジョブの実行

プロジェクトは、シミュレーション・モードまたは実行モードを使用して実行できます。

どちらのモードでも、データ・ターゲットまたはデータ・シンクへのデータの物理的な転送など、プロジェクトに含まれているコンポーネントのすべての機能が実行されます。

シミュレーション・モードでは次のことが実行できます。

- 変更内容を保存せずにプロジェクトを実行する。
- 変換プロセスを順を追ってモニタして検証する。データ・フローはリンク上および含まれているコンポーネント内に表示されます。また、コンポーネントを検査し、マッピングと計算を変更できます。

  変更後、新しい設定でコンポーネントを再初期化し、次のコンポーネントに移ることができます。いずれかのコンポーネントを変更した後、プロジェクトの最初からシミュレーションを開始する必要はありません。

プロジェクトは対話型または非対話型の方法でシミュレートできます。どちらの方法でも、データはシミュレーションの最後にデータ・シンクに書き込まれます。詳細については、「プロジェクトの対話型シミュレーション」(33 ページ) および「プロジェクトの非対話型シミュレーション」(34 ページ) を参照してください。

注意 プロジェクトをシミュレーション・モードで実行する場合、その目的が変換規則のテストであれば、テスト・データ・ターゲットを使用できます。

実行モードでは次のことが実行できます。

- リポジトリに保存されていたプロジェクトとジョブを実行する。保存されていない変更は実行されません。
- プロジェクトを直接実行し、データ・シンク内の変更を反映する。変換プロセスを順を追ってモニタすることはできません。

注意 プロジェクトとジョブは、Sybase ETL Development から実行するか、スケジュール・タスクとして実行できます。詳細については、「ジョブとスケジュール・タスクの管理」(66 ページ) を参照してください。

プロジェクトのカスタマイズ

データ変換プロジェクトは、プログラミング・コードや SQL 文を 1 行ずつ手動で入力しなくても作成できます。たとえば、次の方法を使用できます。

- Query Designer を使用して、クエリ、検索定義、前処理 SQL、後処理 SQL の内部に select 文を生成する。
- コンポーネント間のリンクのデータ・マッピング機能を使用して、データ・ソースとデータ・シンク間で属性をマッピングする。
- 使用しているコンポーネントの組み込み Create Table From Port コマンドを使用して、テンポラリ・ステージング・テーブルまたは永続ステージング・テーブルを作成する。
- 使用しているコンポーネントの組み込み Create Table From Port コマンドを使用して、送信先データベース内にテーブルを作成する。
- Content Explorer を使用して、接続されているすべてのデータ・ソースのスキーマ情報とデータ内容をブラウズする。

- XML via SQL Data Provider コンポーネントを使用して、階層的な XML ドキュメントを読み込み、関係構造を自動的に生成する。
- Sybase ETL Development 内でプロジェクトの実行をスケジュールし、ジョブを作成する。

また、複雑なデータ変換条件を扱う場合は、次の方法を使用できます。

- 手動で最適化された SQL select 文を使用して、データ抽出プロセスを調整する。
- SQL を使用して、前処理コマンドと後処理コマンドの内部でデータ操作コマンドを適用する。
- JavaScript を使用して、プロシージャの記述、複雑な計算の実行、またはオペレーティング・システム環境でのオブジェクトの操作を行う。
- 式内の間接指定 (角カッコ表記) を使用して、環境変数またはユーザ変数を使用してプロジェクトを制御する値を動的に割り当てる。

コンポーネント

レコードごとのコンポーネントのステップ実行

シミュレーション・モードでは、多くの Transformation コンポーネントで現在のデータ・セットを 1 ステップずつ実行し、適用された変換の結果をすぐに視覚化できます。

適合性のあるポート構造とマッピング

プロジェクト内のすべてのデータは、IN ポートと OUT ポートというコンポーネント・ポートを通過します。各ポートには、データ・フローの構造が設定されています。すべてのコンポーネントのポート構造を変更して、構造がコンポーネントの設定に直接依存しないようにすることができます。ポート構造に追加された属性は、コンポーネント内ですぐに参照できます。

コンポーネントを接続するとき、Sybase ETL は OUT ポートと IN ポート間に標準のマッピングを作成しようとします。接続のマッピングは、[Mapping] ウィンドウで変更できます。[Mapping] ウィンドウを開くには、接続リンクを右クリックし、[Mapping] を選択します。詳細については、「現在のマッピングの表示」(35 ページ) を参照してください。

リポジトリ

リポジトリには、Sybase ETL のオブジェクト、プロジェクト、ジョブに関連するすべてのデータと情報が格納されています。

セッションの間、複数のリポジトリに並行してアクセスできます。プロジェクトはリポジトリ間でコピーおよび転送できるので、運用リポジトリを開発リポジトリから切り離すことができます。「プロジェクトの管理」(29 ページ) を参照してください。

通常、リポジトリは部門や会社のような単一のクライアントに属します。複数のクライアントが同じリポジトリを使用することもできます。各クライアントは任意の数のクライアント・ユーザをサポートでき、各ユーザは情報へのアクセスを制御するユーザ名とパスワードを持ちます。

注意 リポジトリ・テーブル内のデータを手動で操作しないでください。リポジトリを手動で操作した場合、Sybase ではそのリポジトリの機能を保証できません。また、リポジトリが使用できなくなり、作業内容が失われる場合があります。

データ型とデータ・フォーマット

データ・ソースのデータ型は変換時に維持されます。

Sybase ETL では、文字列データ型と数値データ型が内部で識別されます。データ・プロバイダまたはデータ・シンクの [Standardize Data Format] オプションを使用すると、データが標準フォーマットに自動的に変換され、続いてターゲット・データベースで処理できるフォーマットに変換されます。異なるデータベースで作業している場合、さまざまな日付フォーマットや数値フォーマットを手動で変換する必要はありません。

デフォルトでは、[Standardize Data Format] オプションが選択されています。ただし、日付フィールドや数値フィールドに問題がある場合は、[Preference] ウィンドウでこの設定を無効にし、データを手動で変換できます。詳細については、「設定のカスタマイズ」(22 ページ) を参照してください。

SQL

データ・プロバイダによって渡されるデータの大部分は、Query プロパティに格納されている SQL 文を使用して定義されます。Sybase ETL では、SQL92 規格の修正版をサポートしています。

SQL 文は、手動で記述するか、既存のプロジェクトから Query プロパティにコピーできます。SQL92 の詳細を使用して作業しない場合は、Query Designer を使用してクエリを引き出し、自動的に SQL 文を生成します。

ツール

接続されているすべてのデータ・ソースからの構造情報とカタログ情報には、Sybase ETL のツールからアクセスできます。これらのツールを使用すると、スキーマ情報やデータをブラウズできるだけでなく、新しいデータベース・オブジェクトを作成することもできます。さまざまな Sybase ETL ツールの詳細については、「高度な概念とツール」(59 ページ) を参照してください。

Unicode サポート

すべてのコンポーネントは、Unicode とマルチバイト・データを処理およびサポートするように設計されています。計算、スクリプト、プロシージャで、Unicode 対応の変換機能を使用できます。Sybase ETL の Unicode サポートのレベルでは、次のような抽出、変換、ロードが可能です。

- Unicode 文字
- コンポーネント・プロパティ内の次の Unicode 文字
  - ファイル名またはディレクトリ名
  - テーブル名や属性名などのメタデータ
  - データベース、スキーマ、ユーザ、パスワードなどの接続設定
  - 変換規則
  - 角カッコ表記 (SBN) 式

式

角カッコ表記 (SBN) は、Sybase ETL 環境内で広範に適用される間接指定メカニズムです。角カッコ表記は、式、SQL 文、ファイル名指定の内部で適用できます。角カッコ表記を使用すると、実行時に値を動的に計算したり割り当てたりすることができます。

第 2 章

使用開始にあたって

Sybase ETL の起動

1 Windows で、[スタート] - [プログラム] - [Sybase] - [Sybase ETL Development 4.8] - [Sybase ETL Development] を選択します。

ログイン・ウィンドウが表示されます。

- 接続 ― `Repository`
- クライアント ― `transformer`
- クライアント・ユーザ名 ― `TRANSFORMER`
- パスワード ― `transformer`

これらの値は、初回ログイン時に自動的に設定されます。以降のログインで、この情報を選択または入力する必要が生じる場合があります。

[Logon] をクリックします。

2 ナビゲータで、
[Repository] - [*TRANSFORMER.transformer.Repository*] - [Projects] をクリックして、利用可能なプロジェクトのリストを開きます。

注意 プロジェクトのリストには、製品に同梱されているデモ・プロジェクトが表示されます。各デモ・プロジェクトには、コンポーネントの使用方法やシナリオの実装方法を示すサンプルが入っています。

3 既存のプロジェクト名をダブルクリックして開くか、[Projects] を右クリックし、[New] を選択して新しいプロジェクトを作成します。

デモ・リポジトリの新規ユーザ・アカウントの設定

デモ・リポジトリに新規ユーザを追加するには、次の手順に従います。

1 Sybase ETL Development のインタフェースから [File] - [Open Repository] を選択します。

2 [Client User] フィールドに新しいクライアント・ユーザ名を入力します。

注意 デモのプロジェクトやジョブにアクセスする場合は、クライアント名を変更しないでください。

3 パスワードを入力します。

4 [Register new user] オプションを選択します。

5 [Show all objects] オプションを選択します。このオプションを選択しなければ、デモのプロジェクトやジョブにアクセスできません。

6 [Logon] をクリックします。

7 パスワードを再入力し、[OK] をクリックします。

Sybase ETL Development インタフェースの操作

Sybase ETL Development のインタフェースは、次の要素で構成されています。

- 設計ウィンドウ　－　変換規則に基づいてプロジェクトやジョブを作成するために使用します。
- コンポーネント・ストア － プロジェクトのコンポーネントを検索するために使用します。
- ナビゲータ － プロジェクト、ジョブ、テンプレートを検索するために使用します。最近アクセスしたプロジェクト、ジョブ、テンプレートもここに表示されます。
- アシスタント・ウィンドウ － 現在のタスクに関するヘルプが表示されます。
- プロパティ・ウィンドウ － コンポーネントのプロパティを設定するために使用します。

ナビゲータ

ナビゲータを使用すると、次の操作を実行できます。

- リポジトリの管理
- リポジトリ内の移動および参照
- プロジェクトとジョブの管理
- プロジェクトとジョブの実行
- ユーザ・アカウントの管理

リポジトリの管理

Sybase ETL のリポジトリは、プロジェクト、ジョブ、セッション・パラメータに関連するすべてのデータを保存および管理するテーブルの集まりです。Sybase IQ、Microsoft SQL Server、Sybase Adaptive Server® Anywhere (ASA)、および Microsoft Access データベースを Sybase ETL リポジトリとして使用できます。

注意 リポジトリ・テーブルのデータを手動で操作しないでください。リポジトリが使用できなくなったり、データが消えたりする可能性があります。リポジトリを手動で操作した場合、Sybase ではそのリポジトリの機能を保証できません。

プロジェクトまたはジョブにアクセスするには、それぞれのリポジトリにログオンする必要があります。リポジトリを開くには、少なくとも 1 つのクライアントと 1 つのクライアント・ユーザを割り当てる必要があります。1 つのクライアントに複数のクライアント・ユーザを割り当てることもできます。

❖ リポジトリを開く

1 [File] - [Open Repository] を選択します。または、ナビゲータでリポジトリを右クリックし、[Open Repository] を選択します。

2 接続リストからリポジトリを選択し、[Logon] をクリックします。

❖ リポジトリ接続の終了

1 ナビゲータでリポジトリ名を右クリックし、[Close Connection] を選択します。

2 確認ダイアログで [Yes] をクリックして、接続と開いているすべてのプロジェクトおよびジョブを終了します。

リポジトリを終了すると、現在リポジトリに接続しているユーザ・セッションがすべて終了します。

❖ **クライアント・ユーザ・セッションの終了**

1 ナビゲータでリポジトリ名を右クリックし、[Close Client] を選択します。

2 確認ダイアログで [Yes] をクリックして、クライアントと開いているすべてのプロジェクトおよびジョブを終了します。

❖ **リポジトリの追加**

1 [File] - [Open Repository] を選択して [Repository Logon] ウィンドウを開きます。

2 [Add] をクリックします。

3 新しいリポジトリ接続のパラメータを入力し、[Save] をクリックします。

新しいリポジトリにアクセスするには、少なくとも 1 つのクライアントと 1 つのクライアント・ユーザ定義を作成する必要があります。

❖ **クライアントとクライアント・ユーザの作成**

1 [Repository Logon] ウィンドウで、[Client] フィールドにクライアント名を入力します。

2 [Client User] フィールドにクライアント・ユーザ名を入力します。名前は 255 文字までの英数字とし、先頭に数字を使用することはできません。

3 パスワードを入力します。

4 [Register New User] オプションを選択します。

5 クライアント内の既存のプロジェクトをすべて表示する権限がクライアント・ユーザにある場合は、[Show All Objects] オプションを選択します。

6 [Logon] をクリックします。

7 パスワードを再入力し、[OK] をクリックします。

注意 [Use Accounts] ウィンドウからもユーザを作成できます。「ユーザの作成」(18 ページ) を参照してください。

❖ **リポジトリの編集**

1 [File] - [Open Repository] を選択して [Repository Logon] ウィンドウを開きます。

2 修正するリポジトリを選択し、[Edit] をクリックします。

3 変更を加えたら [Save] をクリックします。

❖ **リポジトリの削除**

1 接続リストからリポジトリを選択し、[Remove] をクリックします。

2 確認ダイアログで [Yes] をクリックします。

リポジトリ内の移動および参照

ナビゲータの階層ツリー・リストには、次の項目が表示されます。

- 開いているリポジトリ。
- 開いているリポジトリに対するクライアント・ユーザ・セッション。
- プロジェクト、ジョブ、テンプレートなど、リポジトリに保存されているオブジェクト。
- 最近開いたプロジェクト、ジョブ、テンプレート。

リポジトリは複数のクライアント・ユーザ・セッションで同時に開くことができます。クライアント・ユーザはクライアントの一部です。クライアント・ユーザもクライアントも、リポジトリにログオンするときに登録されます。

次の例はツリー構造を示しています。

```
Repositories
-- <RepositoryName1>
---- <ClientUser1>.<Client1>.<Repository Name1>
------ Recent
-------- Recently opened projects
-------- Recently opened jobs
-------- Recently opened templates
------ Projects
-------- Project_1
-------- Project_2
-------- Project_N
------ Jobs
-------- Job_1
-------- Job_2
-------- Job_M
------ Templates
-------- Template_1
-------- Template_L
---- <ClientUser1>.<ClientM>.<Repository Name1>
---- <ClientUserN>.<Client1>.<Repository Name1>
-- <RepositoryName2>
```

プロジェクトとジョブの管理

ナビゲータからプロジェクトとジョブを管理できます。「第 3 章 プロジェクトとジョブ」を参照してください。

ユーザ・アカウントの管理

ナビゲータから次の操作を行うことができます。

- ユーザを作成する。
- ユーザを削除する。
- パスワードを変更する。
- 可視性を変更する。

登録済みのクライアント・ユーザのみがリポジトリにアクセスできます。クライアント・ユーザの登録は [Repository Logon] ウィンドウまたは [User Accounts] ウィンドウで行うことができます。

❖ **ユーザの作成**

1 ナビゲータでリポジトリ名を右クリックし、[User Accounts] を選択します。

2 [Add User] をクリックします。

3 ユーザ名を入力します。ユーザ名の条件は次のとおりです。

- 英数字のみを使用する。
- 先頭は英字にする。
- 最大 255 文字まで。
- 空にしない。

4 パスワードを入力します。

5 パスワードを再入力します。

6 他のリポジトリ・ユーザに属するオブジェクトを表示するには、[Show All Objects] オプションを選択します。

7 [OK] をクリックします。

❖ **ユーザの削除**

1 ナビゲータでリポジトリ名を右クリックし、[User Accounts] を選択します。

2 削除するユーザを選択し、[Remove User] をクリックします。

選択したユーザが現在リポジトリに接続している場合は、ユーザを削除して開いているすべてのプロジェクトおよびジョブを終了するかどうかを確認するプロンプトが表示されます。[Yes] をクリックします。

3 削除するユーザのパスワードを入力し、[OK] をクリックします。

❖ **パスワードの変更**

1 ナビゲータでリポジトリ名を右クリックし、[User Accounts] を選択します。

2 パスワードを変更するユーザを選択します。

3 [Change Password] をクリックします。

4 ユーザの既存のパスワードと新しいパスワードを入力します。新しいパスワードを再入力します。

5 [OK] をクリックします。

プロパティ・ウィンドウ

プロパティ・ウィンドウでは、次の操作を実行できます。

- コンポーネントのプロパティを表示して編集する。
- コンポーネントの必須プロパティを確認する。
- コンポーネントのプロパティの動的評価を許可する。
- コンポーネントのプロパティを暗号化する。
- コンポーネントにカスタム・プロパティを追加してその値を編集する。
- コンポーネント設定ウィンドウにアクセスする。

コンポーネント固有のプロパティ設定については、「第 5 章 コンポーネント」を参照してください。

コンポーネントのプロパティの表示と編集

コンポーネントのプロパティおよび値を表示して編集するには、設計ウィンドウでコンポーネントを選択します。選択したコンポーネントのすべてのプロパティが、プロパティ・ウィンドウに表示されます。

必須プロパティの確認

プロパティ・ウィンドウに**太字**で表示されるプロパティ名は、コンポーネントが正しく動作するために必要なプロパティであることを示します。その他のプロパティは省略可能で、コンポーネントの微調整や設定に使用できます。

動的な式の許可

動的な間接式 (SBN 式) の評価を許可するには、[Evaluate] オプションを選択します。角カッコ [ ] を使用して、対応するフィールドに SBN 式を入力します。[Evaluate] オプションを使用すると、設計時に静的な値を割り当てるのではなく、実行時に動的なプロパティ設定を計算して評価できます。

一部のプロパティ項目については、[Evaluate] オプションがデフォルトで選択されています。

❖ **プロパティ評価の有効化／無効化**

1 プロパティ・ウィンドウで、実行時に評価するプロパティ名を右クリックします。

2 [Evaluate] を選択します。

プロパティの暗号化

プロジェクトおよびジョブのデータとプロパティの値は、Sybase ETL リポジトリに保存されています。Sybase ETL リポジトリ内のほとんどのレコードは暗号化されず、判読可能な形式で表現されます。パスワード・プロパティについては、[Encrypt] オプションがデフォルトで選択されています。

❖ **プロパティ値の暗号化**

1 プロパティ・ウィンドウで、プロパティ名を右クリックします。

2 [Encrypt] をクリックします。

または、プロパティ値の横にある [Encrypt] チェックボックスをオンにします。

カスタム・プロパティの追加と編集

カスタム・コンポーネント・プロパティを追加または編集するには、プロパティ・ウィンドウを使用します。他のプロパティと同様に、カスタム・プロパティにも、式やユーザ定義プロシージャで参照できる変数が組み込まれます。

詳細については、「カスタム・プロパティ」(100 ページ) を参照してください。

コンポーネント設定ウィンドウへのアクセス

プロパティ・ウィンドウで、プロパティ・アイコンをクリックして、選択したコンポーネントの設定ウィンドウを開きます。

注意 設定ウィンドウがないコンポーネントもあります。

設計ウィンドウ

設計ウィンドウでは、次の操作を実行できます。

- プロジェクトとジョブを作成および修正する。詳細については、「プロジェクトとジョブ」(29 ページ) を参照してください。
- プロジェクトをシミュレートおよび実行する。詳細については、「プロジェクトのシミュレーション」(31 ページ) と「プロジェクトの実行」(41 ページ) を参照してください。
- ジョブを実行する。「ジョブの管理」(42 ページ) を参照してください。

❖ 設計ウィンドウへのコンポーネントの追加

プロジェクトまたはジョブを作成するには、コンポーネントを追加して接続し、そのプロパティを設定する必要があります。コンポーネントは次の 4 つの方法で設計ウィンドウに追加できます。

1. コンポーネント・ストアでコンポーネントを選択し、設計ウィンドウにドラッグします。この方法では、プロジェクトだけでなく、ジョブにもコンポーネントを追加できます。その他の方法では、プロジェクトにのみコンポーネントを追加できます。
2. コンポーネント・ストアでコンポーネントをダブルクリックします。
3. コンポーネント・ストアでコンポーネントを右クリックし、[Add] を選択します。
4. 設計ウィンドウで既存のコンポーネントのポートを右クリックし、[Add Component] を選択します。コンポーネント・タイプをポイントし、追加するコンポーネントを選択します。選択したポートが入力ポートであるか出力ポートであるかに応じて、既存のコンポーネントの前後にコンポーネントが追加されます。

❖ 設計ウィンドウからのコンポーネントの削除

1. 設計ウィンドウで、削除するコンポーネントを選択します。
2. 右クリックして [Delete] を選択します。

❖ **一般的なコマンドの処理**

1 設計ウィンドウ内を右クリックして、通常のプロジェクトを開きます。このメニューには、[Close] や [Print] などの一般的なコマンドが表示されます。コンポーネントを右クリックして、[Component] ポップアップ・メニューを開きます。[Component] メニューには、コンポーネント固有のコマンドが表示されます。

2 実行するアクションを選択します。

コンポーネント・ストア

コンポーネント・ストアは、コンポーネントを一般的な目的によって分類する複数のセクションで構成されています。

コンポーネント・ストアからプロジェクトへコンポーネントを追加するには、次のいずれかの操作を行います。

- コンポーネントを設計ウィンドウにドラッグします。
- コンポーネントを右クリックし、[Add] を選択します。
- コンポーネントをダブルクリックします。

設定のカスタマイズ

Sybase ETL Development で設定グループをカスタマイズするには、[Preferences] ウィンドウを使用します。

- Workbench
  - Appearance
  - Data Viewer
  - Query Designer
- Engine
- Performance Logging

❖ 設定のカスタマイズ

1　Sybase ETL Development のメイン・ウィンドウから [File] - [Preferences] を選択します。

2　[Preferences] ウィンドウで、[Appearance] を選択し、次のオプションを設定します。

- [Locale for user interface display] － 使用している環境のロケール言語を選択します。_de (ドイツ語)、_en_US (米語)、または _en_GB (英語) を選択できます。デフォルトは _en_US です。
- [Show assistant for creating projects] － このオプションを選択すると、プロジェクトを完成させるプロセスを案内したり、開いているプロジェクトの現在の状態に関する情報を確認したりするためのアシスタントが表示されます。
- [Default font for displaying text] － [Text Data Provider] および [Text Data Sink] コンポーネント・ウィンドウにテキスト・ファイルの内容を表示するためのフォントを選択します。この設定は、UNICODE など西欧言語以外の文字セットで作業する場合に便利です。デフォルト・フォントは Monospaced です。テキスト表示に推奨されるフォントは Dialog または Monospaced です。
- [Default font for displaying data] － アプリケーション全体でポート・データを表示するためのフォントを選択します。この設定は、UNICODE など西欧言語以外の文字セットで作業する場合に便利です。Sybase ETL Development を初めてインストールする場合、デフォルト・フォントは Dialog になります。以前に Sybase ETL Development をインストールしたことがある場合、フォントは以前に定義した値に設定されます。フォントを Dialog に設定することをおすすめします。

  注意　[Default font for displaying text] および [Default font for displaying data] 設定を使用する場合は、東アジア言語用のファイルをインストールすることをおすすめします。Windows の [コントロール パネル] で [地域と言語のオプション] をクリックし、[言語] タブを選択して、[東アジア言語のファイルをインストールする] オプションを選択します。このオプションを有効にすると、Unicode 文字の表示に必要なフォントがインストールされます。

- [Create a new project on startup] － このオプションを選択すると、Sybase ETL の起動時に新規プロジェクトが自動的に作成されます。

- [Create links automatically when components are added] － このオプションを選択すると、既存のコンポーネントと、プロジェクトに追加された新しいコンポーネントの間に、リンクが自動的に作成されます。
- [Default action on double-clicking a connection] － シミュレーション中に接続をダブルクリックしたときに [Mapping] ウィンドウを開くか [Preview] ウィンドウを開くかを指定します。[Mapping] ウィンドウにはマッピング情報、[Preview] ウィンドウには接続データのプレビューが表示されます。デフォルトは [Mapping] です。
- [Display qualified transformation objects] － このオプションを選択すると、ナビゲータ内のオブジェクト名の先頭に所有者名が付きます。たとえば、このオプションを選択すると、プロジェクト名は次のようにナビゲータに表示されます。

  ```
  TRANSFORMER.Demo Character Mapper
  ```

  ここで、TRANSFORMER は、プロジェクトを作成したクライアント・ユーザの名前です。
- [Use unique object names] － このオプションを選択すると、リポジトリの接続時に固有のプロジェクト名とジョブ名が適用されます。
- [Show password in component tooltips] － 基になるデータベースへのログインに使用されるパスワードをコンポーネントのヒントに表示する場合は、このオプションを選択します。デフォルトでは、パスワードは一連のアスタリスクとしてヒントに表示されます。
- [Use enhanced color accessibility] － 色の識別が困難なユーザに対応するためにコンポーネントのポートの色を変更するには、このオプションを選択します。このオプションを選択すると、ポートのデフォルトの色が黄色から青色に変わるので、色の識別が困難なユーザでもポートの状態を見分けることができます。
- [Use vertical component layout] － プロジェクトとジョブの配列をデフォルトの左から右ではなく、上から下へと縦に表示するには、このオプションを選択します。
- [Show information dialogs] － アクションを実行するときに情報プロンプトが表示されないようにする場合は、このオプションの選択を解除します。

- [Number of recently opened projects, jobs, and templates to display] − 最近アクセスしたプロジェクト、ジョブ、テンプレートをナビゲータと [File] メニューに表示する数を指定します。
- [Open the last repository automatically] − このオプションを選択すると、再起動時に、前回ログオンしたリポジトリに自動的に接続します。
- [Save repository client password] − リポジトリにログインした後にクライアント・パスワードを保存する場合は、このオプションを選択します。選択した場合は、次のログイン時にパスワードを入力する必要がなくなり、[Repository] ウィンドウの [Password] フィールドに自動的に入力されます。

3 [Data Viewer] を選択し、エクスポートされたデータ・フィールドの最大長を指定します。デフォルト値は 255 です。ここで指定した値より長いデータ・フィールドは、エクスポート・ファイルで切り捨てられます。

4 [Query Designer] を選択し、次のオプションを設定します。

- [Enable delete functionality of database objects] − このオプションを選択すると、Query Designer でテーブルを右クリックして [Truncate Object] を選択したときに、そのテーブルのすべてのレコードが削除されます。
- [Default number of records to retrieve from the Query Designer] − Query Designer によって取得されるデータ・レコードのデフォルト数を指定します。デフォルトは 25 です。
- [Create joins automatically] − テーブルまたはビュー内で使用されている同じ属性名の間に自動的にジョインを作成するには、このオプションを選択します。
- [Use brackets when creating joins] − 生成されたクエリでジョイン句を自動的にカッコで囲むには、このオプションを選択します。たとえば、select 文は次のように表示されます。

```
select * FROM SALES ((<join statement 1>)
<join statement 2>)
```

- [Default number of recently accessed tables and views to display] − 最近アクセスしたテーブルおよびビューを Query Designer の [Recent] タブに表示する数を指定します。デフォルト値は 25 です。

5 [Engine] を選択し、次のオプションを設定します。

- [Start local engine during application startup] − Sybase ETL の起動時にローカル・エンジンを開始するには、このオプションを選択します。
- [Interval between engine monitor updates (sec)] − Engine Monitor の更新間隔を秒数で指定します。デフォルトは 5 秒です。
- [Rate of simulation (msec)] − シミュレーションのトレース遅延を設定してシミュレーション速度を制御します。シミュレーションのトレース遅延オプションには、10 ～ 9999 ミリ秒の値を設定できます。デフォルトは 250 ミリ秒です。
- [Grid engine ping timeout (sec)] − ローカル・グリッド・エンジンを開始または再開する前にグリッド・エンジンへのアクセスが許可されている時間を秒数で指定します。デフォルトは 60 秒です。
- [Interval between progress monitor updates (sec)] − ジョブ実行の進行状況モニタの更新間隔を秒数で指定します。デフォルトは 5 秒です。
- [Allow selection of the execution engine] − プロジェクトの実行に使用するグリッド・エンジンを指定できるようにする場合は、このオプションを選択します。
- [Execution engine server] − プライマリ・グリッド・エンジン・サーバの IP アドレスを指定します。
- [Execution engine port] − プライマリ・グリッド・エンジン・サーバのポート・アドレスを指定します。

6 [Performance Logging] を選択し、パフォーマンス・データへのロギングの詳細レベルを指定します。

- 0 − パフォーマンス・データをリポジトリに記述しない。
- 1 − パフォーマンス・データをリポジトリに記述する。これはデフォルトの値。

7 [Save] をクリックします。変更を有効にするには、Sybase ETL の再起動が必要になる場合があります。再起動しなければ、変更は次回 Sybase ETL を起動したときに有効になります。

トラブルシューティング

インストール時に、Sybase ETL インストーラによってデータ・ソースの初期セットが作成されます。これらのリポジトリ・データ・ソースが何らかの理由で失われた場合、それらを復元しなければ Sybase ETL が開きません。デモ・リポジトリの ODBC データ・ソースの初期セットを復元するには、次の手順に従います。

1　ODBC ユーザ・データ・ソースを設定します。

   a　[スタート] - [設定] - [コントロール パネル] - [管理ツール] - [データ ソース (ODBC)] を選択します。

   b　[Add] をクリックします。

   c　リストから [Microsoft Access Driver] を選択します。[Finish] をクリックします。

   d　[データ ソース名] フィールドに DEMO_Repository と入力します。

   e　インストール・ディレクトリの *Demodata* フォルダから *IQETLDEMO_REP.MDB* データベースを選択します。[OK] をクリックします。

2　[Repository Logon] ウィンドウでリポジトリの接続を設定します。

   - [File] - [Open Repository] を選択して [Repository Logon] ウィンドウを開きます。
   - 接続リストからリポジトリを選択し、次のいずれかを選択します。
     - [Edit]
     - [Add and enter a name for the connection]
   - インタフェース・リストから [ODBC] を選択します。
   - ホスト・リストから、[DEMO_Repository] を選択します。
   - [Save] をクリックします。

3 デモ・リポジトリで、プロジェクトに必要な次の ODBC ユーザ・データ・ソースを追加設定します。

- ドライバ − Microsoft Access
- 名前 − ETLDEMO_DWH、データベース − DEMO_DWH.MDB
- 名前 − ETLDEMO_GER、データベース − DEMO_GER.MDB
- 名前 − ETLDEMO_US、データベース − DEMO_US.MDB

これらのユーザ・データ・ソースのデータベース・ファイルは、インストール・ディレクトリの *Demodata* フォルダにもあります。

第 3 章

プロジェクトとジョブ

プロジェクトの管理

プロジェクトは、コンポーネントと、ポートを介してコンポーネントを接続するリンクで構成されています。プロジェクトに関連した基本操作には、作成、削除、名前の変更、保存などがあり、複雑な操作には、シミュレーションなどがあります。

Sybase ETL プロジェクトは、1 つまたは複数の Source コンポーネントで始まり、1 つまたは複数の Destination コンポーネントで終わります。

Sybase ETL のコンポーネントは次のとおりです。

- Data Provider コンポーネントは通常、Transformation コンポーネント、Processing コンポーネント、または Data Sink コンポーネントに接続されています。
- Transformation コンポーネントと Processing コンポーネントには入力ポートと出力ポートがあり、他の種類のコンポーネントと隣接させることができます。
- Transformation コンポーネントで複数の入力データ・ストリームが許可されている場合は、始点となる複数の Source コンポーネントが必要です。
- Transformation コンポーネントに複数のデータ・ストリーム出力がある場合は、各データ・ストリームをコンポーネントに接続できます。

❖ **プロジェクトの作成**

1 ナビゲータで [Projects] を右クリックし、[New] - [Projects] を選択します。または、[File] メニューの [New] - [Projects] を選択します。

2 コンポーネント・ストアから設計ウィンドウに、プロジェクトのコンポーネントをドラッグします。

❖ **プロジェクトの変更**

1 ナビゲータで、変更するプロジェクトをダブルクリックします。

2 変更を加えて、プロジェクトを保存します。

❖ **プロジェクトのロック解除**

別のユーザ・クライアントによってロックされているプロジェクトは、read-only モードで開きます。

- プロジェクトを読み込みまたは書き込みできるようにするには、ロックされたプロジェクトを開いたときに表示されるウィンドウで [Unlock and Open] をクリックします。

❖ **プロジェクトのコピー**

1 コピーするプロジェクトをダブルクリックして、設計ウィンドウで開きます。

2 ナビゲータでプロジェクトを右クリックし、[Save As] を選択します。または、[File] メニューの [Save As] を選択します。

複数のリポジトリで作業している場合は、ターゲット・リポジトリを選択します。

3 新しいプロジェクトの名前を入力します。既存のプロジェクトのコピーが作成されます。元のプロジェクトはそのまま残り、元のプロジェクトへの参照も保存されません。

❖ **プロジェクトの転送**

複数のリポジトリを使用している場合、あるリポジトリから別のリポジトリにプロジェクト全体をコピーして、元のプロジェクトへの参照を保持できます。たとえば、開発リポジトリのプロジェクトをテスト・リポジトリまたは運用リポジトリに移動する場合があります。元のプロジェクトへの参照を保存することにより、転送の次回開始時にプロジェクトが認識されて、受信オブジェクトに関連するすべてのものが選択的に置き換えられます。

1　ナビゲータで転送するプロジェクトを右クリックし、[Transfer] を選択します。または、[File] メニューの [Transfer] を選択します。

2　プロジェクトを転送するリポジトリを選択します。

注意 転送によってコピーされるのは、プロジェクトの定義と実行プロパティです。パラメータ・セットなどの関連データは転送されません。

❖ **プロジェクトの削除**

1　ナビゲータでプロジェクトを右クリックし、[Delete] を選択します。

2　[Yes] をクリックして、削除を確定します。

❖ **プロジェクト名の変更**

1　ナビゲータでプロジェクトを右クリックし、[Rename] を選択します。

2　プロジェクトの新しい名前を入力し、[OK] をクリックします。

❖ **実行プロパティのリセット**

インクリメンタル・ロードのロード・オプションをリセットするには、次の手順に従います。

1　ナビゲータでプロジェクトを右クリックし、[Reset Execution Properties] を選択します。

2　[Yes]　をクリックして、実行プロパティのリセットを確定します。Load Index Value (DB Index Load コンポーネント) の現在の値がリセットされます。

プロジェクトのシミュレーション

プロジェクトをシミュレートすると、変換プロセスを順を追ってモニタして検証できます。プロジェクトの実行とは対照的に、シミュレーションでは次のことができます。

- 変更内容を保存せずにプロジェクトを実行する。
- 変換プロセスのすべての段階でデータを表示する。

シミュレーションの最後には、データがデータ・シンクに書き込まれます。Data Calculator など多くの Transformation コンポーネントでは、シミュレーション中に変換規則とサンプル値を変更して、潜在的なすべてのコンテンツの規則ベースを検証できます。

次のいずれかの方法で、プロジェクトをシミュレートできます。

- 対話型 － [Run] - [Start] を選択してシミュレーション対象のプロジェクトを初期化し、[Run] - [Step] を選択して手動でコンポーネントを進め、プロジェクトの実行をシミュレートします。

  または、[Run] - [Interactive Trace] を選択してシミュレーション対象のプロジェクトを初期化し、事前に定義したペースでコンポーネントを自動的にステップ実行します。

- 非対話型 － [Run] - [Noninteractive Trace] を選択して、対話なしでプロジェクトをシミュレートします。「プロジェクトの非対話型シミュレーション」(34 ページ) を参照してください。

注意 すべてのコンポーネントが正常に初期化された後にのみ、プロジェクトをシミュレートできます。

シミュレーションの基本機能は、次の高度な手順で構成されています。

- 対話型または非対話型のシミュレーションを開始する。
- コンポーネントをステップ実行する。
- 接続リンクまたはコンポーネント内のデータ・フローを表示する。
- コンポーネントを変更し、再初期化して、データ・フローのシミュレーションを続行する。

シミュレーションの詳細レベルでは、次のことができます。

- 接続リンクのデータ・コンテンツを表示する。
- コンポーネント内の入力データと出力データを表示する。
- プロパティまたは計算を変更し、変換規則とサンプル値を変更して規則ベースを検証できるようにする。
- 計算またはプロパティを変更した後で、再度コンポーネントをステップ実行する。
- what-if シナリオを実行する。
- プロジェクトの複数のステップを実行する。

プロジェクトの対話型シミュレーション

プロジェクト全体を対話的に実行するには、[Run] - [Interactive Trace] を選択します。プロジェクトを対話的にシミュレートする場合は、シミュレーションを任意の時点で中止し、[Run] - [Step] または [Run] - [Step Through] を選択して、残りのプロジェクト・コンポーネントを手動でステップ実行することができます。

❖ プロジェクトの対話型シミュレーション

1　シミュレーションを開始するには、ツールバーの [Start] をクリックします。[Start] をクリックすると、次の処理が行われます。

- プロジェクト内のすべてのコンポーネントが初期化されます。
- プロジェクト内のすべての接続が検証されます。
- プロジェクト内のすべての pre-SQL 文が実行されます。
- すべての静的ルックアップ・コンポーネントのデータが取得およびキャッシュされます。シミュレーション中にルックアップ・テーブルのデータに加えられた変更は、シミュレーション・プロセスには反映されません。

2　コンポーネントを選択し、ツールバーの [Step] アイコンをクリックして、コンポーネントを実行します。

コンポーネントのステップ実行とは、1 つのコンポーネントを実行または処理することです。1 つのステップで処理されるデータ・レコードは、コンポーネントの IN ポートに現在入力されているレコードです。

コンポーネントが複数回ステップ実行され、その間に他のコンポーネントがない場合、受信または転送されるレコードの数は一定になります。多くのコンポーネントは、コンポーネント内およびプロジェクト・ビュー外の両方からステップ実行できます。

3　接続リンクまたはコンポーネント内のデータ・フローを表示します。

- 変換プロセスを通じてデータを表示するには、コンポーネント間またはコンポーネントのポート間のリンクを調べます。Data Calculator や Data Splitter などのその他のコンポーネントには、プレビュー機能が組み込まれています。
- 接続リンクでデータを表示するには、右クリックして [Preview] を選択します。

- 現在ポートにあるデータを表示するには、ポートを右クリックして [Preview] を選択します。

  注意 処理レコードがない場合や、使用可能なシミュレーション・データがない場合、[Preview] オプションは無効になります。

- コンポーネント内からデータを表示するには、コンポーネントをダブルクリックするか、プロパティ・ウィンドウの [Rule] アイコンをクリックします。このオプションを使用して、Data Calculator や Data Splitter などのコンポーネント内での変換規則の影響を確認します。

4 コンポーネントを変更し、初期化します。

コンポーネントを変更したら、再初期化してデータ・フローのシミュレーションを続行できます。現在のプロジェクトのシミュレーション全体をやり直す必要はありません。

a コンポーネントをダブルクリックして、プロパティ・ウィンドウでプロパティを変更します。

b 変更内容を保存します。

c コンポーネントを右クリックし、[Initialize] を選択します。

注意 対話型トレースでは、DB コンポーネントでの post-SQL 文の実行などの後処理タスクは実行されません。

❖ **トレース遅延の設定**

シミュレーションの対話型トレース遅延オプションを設定することによって、シミュレーション速度を制御できます。

1 [File] - [Preferences] - [Engine] を選択します。

2 シミュレーション・フィールドの [Rate] の値を変更します。10 ～ 9999 ミリ秒の値を入力できます。デフォルト値は 250 ミリ秒です。

プロジェクトの非対話型シミュレーション

対話なしでプロジェクトをシミュレートするには、[Run] - [Non Interactive Trace] を選択します。非対話型トレースでは、プロジェクトがステップ実行され、実行の最後に接続レコード数が更新されます。

シミュレーションがまだ開始されていない場合は、このオプションを選択すると、シミュレーションが自動的に開始され、すべてのデータを処理するまで継続されます。

対話型トレースとは異なり、非対話型トレースを使用してプロジェクトをシミュレートした場合、シミュレーションを任意の時点で中止することはできません。

現在のマッピングの表示

[Mapping Definition] ウィンドウには、隣接した IN 構造と OUT 構造の属性間の現在のマッピングが表示されます。

現在のマッピングを表示するには、次の手順に従います。

1　接続リンクを右クリックし、[Mapping] を選択します。

2　[Mapping Definition] ウィンドウで、次の手順に従います。

- 接続ポートのすべての属性とその現在のマッピングを表示するには、[Display structure] を選択します。
- フィールドだけでなく、現在のレコードの値も表示する場合は、[Display structure and values] を選択します。このビューには、リンクに接続しているポートの現在の内容が表示されます。ポートにデータが含まれていない場合、ポート構造のみがこのウィンドウに表示されます。ポートに到達するまでプロジェクトをステップ実行して、ポートにデータを入力することができます。

自動マッピングの適用

マッピングを作成するには、[Mapping] メニューから定義済みのマッピング・シーケンスのいずれかを選択します。

- [Create mapping by Order] − IN 構造と OUT 構造のポート属性を順番にマッピングします。双方の属性数が異なる場合、一部のポート属性はマッピングされません。
- [Create mapping by Name] − 名前に従って IN 構造と OUT 構造のポート属性をマッピングします。
- [Create mapping by Name Case Sensitive] − 大文字と小文字を区別した名前に従って IN 構造と OUT 構造のポート属性をマッピングします。
- [Create mapping by Prefix] − 指定されたプレフィクスは無視し、名前に従って IN 構造と OUT 構造のポート属性をマッピングします。
- [Create mapping by Best Match] − よく似た IN 構造と OUT 構造のポート属性をマッピングします。

手動マッピングの適用

1 つのマッピングを手動で作成するには、接続ポイントを選択し、ポート属性の接続ポイントにドラッグします。

現在のマッピングを変更するには、接続ポイントのマッピング・ラインを選択し、マッピングされていないポート属性にドラッグします。

1 つのマッピングを削除するには、マッピングを右クリックし、[Delete] を選択します。

リンクのマッピングをすべて削除するには、[Mapping] メニューの [Remove All] を選択します。

マッピングされた属性の表示

デフォルトでは、[Mapping Definition] ウィンドウに、IN 構造と OUT 構造のすべてのポート属性が表示されます。マッピングされた属性のみを表示するには、ツールバーの [Display only mapped attributes] アイコンをクリックします。

ポート属性の管理

[Structure Viewer] ウィンドウでは、ポート属性の追加と削除や、設定または既存の属性の変更を行うことができます。

❖ **ポート構造への属性の追加**

1 設計ウィンドウでポートを右クリックし、[Edit Structure] を選択します。

2 [Structure Viewer] ウィンドウで [Actions] - [Add] を選択するか、属性を右クリックして [Add] を選択します。

3 属性の名前を入力します。ポート属性の名前は英文字で始める必要があり、英数字のみを使用できます。予約されている JavaScript キーワードは使用しないでください。詳細については、「変数」(70 ページ) を参照してください。

[Populate Attribute] オプションを選択して、複数のポート構造に属性を追加します。選択した接続に参加しているすべてのポート構造に新しい属性が追加され、自動的にマッピングされます。[OK] をクリックします。

4 その他の詳細を指定します。

❖ **ポート構造からの属性の削除**

1　設計ウィンドウでポートを右クリックし、[Edit Structure] を選択します。

2　[Structure Viewer] ウィンドウで [Actions] - [Remove] を選択するか、属性を右クリックして [Remove] を選択します。

❖ **ポート属性の変更**

1　設計ウィンドウでポートを右クリックし、[Edit Structure] を選択します。

2　[Structure Viewer] ウィンドウで属性設定を変更し、[Save] をクリックします。

詳細については、「ポート構造の管理」(101 ページ) を参照してください。

データ型の変更

レコード構造のデータ型を変更すると、Sybase ETL で変換時にレコード構造に対して使用される内部論理表現が変更されます。送信元テーブルや送信先テーブルのデータ構造定義は変更されません。最終的な Data Sink のデータ構造と生成中の内容との互換性を確保する必要があります。

シミュレーション・フローの表示

シミュレーションを開始した後、次の方法でシミュレーション・フローを参照できます。

- 緑色の点線で囲まれたボックスはアクティブなコンポーネントを示し、コンポーネント間を 1 ステップずつ移動します。
- レコード数がリンク上に表示され、ボックスの移動に従って移動します。

各ステップ内で処理されるレコードの数は、Read Block Size プロパティを持つ前のコンポーネントの Read Block Size の現在値によって異なります。

シミュレーションの実行時には、小さな値を選択すると便利です。Read Block Size を大きな値に設定すると、プロジェクト実行時のパフォーマンスを大幅に高めることができます。

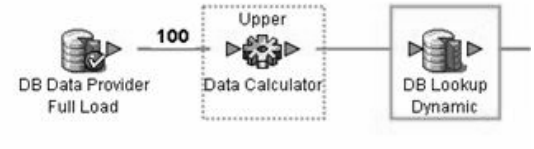
現在のコンポーネントおよび選択したコンポーネントからのステップ実行

シミュレーションを開始すると、最初に実行されるコンポーネントが緑色の点線ボックスに囲まれて表示されます。

変更を行わずにプロジェクトをステップ実行すると、ボックスは成功アイコンまたは失敗アイコンを表示しながらシミュレーションの最後までコンポーネント間を移動します。

現在のコンポーネントとは別のコンポーネントを選択して、プロパティを検査したり変更したりできます。選択したコンポーネントは緑色の実線ボックスで示されます。

図 3-1：現在のコンポーネントと選択したコンポーネント



ツールバーの [Step] をクリックするか、[Run] - [Step] を選択すると、現在のコンポーネントが次に実行されます。シミュレーション中に現在のコンポーネントとは異なるコンポーネントを検査または変更するには、そのコンポーネントをクリックします。選択したコンポーネントが緑色の実線ボックスで強調表示されます。

変更を加えた後、選択したコンポーネントまたは現在のコンポーネントのいずれかからシミュレーションを再開します。

- 選択したコンポーネントからシミュレーションを再開するには、右クリックして [Step] を選択します。
- 現在のコンポーネントからシミュレーションを再開するには、ツールバーの [Step] をクリックします。

注意 未処理のコンポーネントを右クリックし、[Initialize and Step] を選択すると、そのコンポーネントは初期化されてステップ実行され、次に処理されるコンポーネントが強調表示されます。

コンポーネントの転送と後方転送

ボックスで示されるシミュレーションのビジュアル・フローは、ほとんどのプロジェクトで単純明快であり、あるコンポーネントから次のコンポーネントに移動するだけです。ただし、プロジェクト・シミュレーションのフローは必ずしも一方向にのみ進むとは限りません。フローの方向は、プロジェクト内で使用されているコンポーネントによって異なります。

Data Calculator や Character Mapper などのコンポーネントの転送では、多数のレコードを受け取り、そのレコードに変換を適用して、レコードを転送します。1 つのステップで処理されるレコードの数は、前のコンポーネントから受け取ったレコードの数によってのみ決定されます。

その他のコンポーネントでは、前の Read Block Size 設定が上書きされます。Staging コンポーネントは、Data Source コンポーネントのクエリで定義されたデータ・ストリームの結果セット全体に対して動作するように設計されています。結果セット全体が IN ポートに配信されるまで、データ・レコードの処理や転送は行われません。Staging コンポーネントは独自の Read Block Size プロパティを使用して、次のステップで転送されるレコード数を変更します。シミュレーション時の各コンポーネントの動作の詳細については、「第 5 章 コンポーネント」を参照してください。

複数のロケーションからのデータのプレビュー

任意の接続リンク、ポート、またはコンポーネントを右クリックし、[Preview] を選択して [Content Browser] ウィンドウを開くと、選択したロケーションで現在使用可能なデータが表示されます。

注意 処理レコードがない場合や、使用可能なシミュレーション・データがない場合、[Preview] オプションは無効になります。

[Content Browser] ウィンドウには、複数のロケーションから複数のプレビューを同時に表示できるタブがあります。コンポーネントの IN ポートと OUT ポートの内容を並列タブでプレビューすると役に立つことがあります。

表示したデータを定義ファイルに保存するには、ツールバーの [Export data] アイコンをクリックします。データをエクスポートするためのオプションを指定します。

シミュレーション中の部分的な実行または初期化

1 つのコンポーネントに変更を加えた後でシミュレーション全体を再開すると、特に多数のコンポーネントで構成されるプロジェクトに多数の入力レコードが含まれる場合、非常に時間がかかります。また、複雑なシミュレーション・フローの途中にある 1 つのコンポーネントをシミュレーションしたいだけの場合、大きなプロジェクトを 1 ステップずつ進めるのは根気が必要です。プロジェクトの対象位置まで複数ステップを進める場合は、コンポーネントを選択し、[Run] メニューの [Step through] または [Start through] を選択します。

特定のコンポーネントまでのシミュレーション

プロジェクトの途中のコンポーネントから現在のプロジェクトの検証を開始するには、コンポーネントを選択し、[Run] - [Start Through] を選択します。シミュレーションによって現在のプロジェクトが開始され、現在のコンポーネントと選択したコンポーネント間のすべてのコンポーネントが処理された後、選択したコンポーネントが処理されます。

Read/Write Block Size の影響

Read Block Size として入力した値によって、1 つのシミュレーション・ステップ中にコンポーネントでフェッチされるレコードの数が定義されます。Write Block Size を設定すると、書き込まれるレコードの数が定義されます。ほとんどの Data Provider コンポーネントには Read Block Size プロパティがあります。ほとんどの Data Sink コンポーネントでは、Write Block Size をカスタマイズできます。Staging コンポーネントなどの変換コンポーネントでは、読み込みと書き込みの両方の値をカスタマイズできます。

注意 Block Size プロパティは、プロジェクトのシミュレーション中と、プロジェクトおよびジョブの実行中に評価されます。シミュレーションの目的には小さな値が適していますが、[Run] - [Execute] をクリックすると、実行速度が低下します。シミュレーションでの Block Size は 32K に制限されています。

複数のデータ・ストリームの制御

ほとんどのプロジェクトは、リンクで接続されたコンポーネントの 1 つのストリームで構成されていますが、接続されていない複数のデータ・ストリームを持つ 1 つのプロジェクトを設定することもできます。Sybase ETL は並列システムなので、ストリームの処理順序を予測することはできません。

複数のデータ・ストリームを使用する場合は、データ・ストリームごとにプロジェクトを設計して、プロジェクト内のすべてのコンポーネントが相互接続するようにすることをおすすめします。このような設計ガイドラインに従うと、プロジェクトを接続してジョブ・プロセス・フローを形成することによってデータ・ストリームを制御できます。

プロジェクトの実行

次のいずれかの方法に従って、デフォルト・グリッド・エンジンでプロジェクトを実行できます。

- [Run] - [Execute] を選択するか、ツールバーの [Execute] アイコンをクリックします。設計ウィンドウで現在開かれているプロジェクトが実行されます。

  実行は、シミュレーションの状態に影響を与えます。保存されていないプロジェクトを実行しようとすると、プロジェクトの保存を求めるメッセージが表示されます。プロジェクトを保存すると、プロジェクトのシミュレーション・データはすべて失われ、プロジェクトが実行されます。

  注意 プロジェクト定義はリポジトリから読み込まれるため、実行前にプロジェクトに変更を保存する必要があります。変更を保存しなければ、実行は開始されません。

- ナビゲータで、実行するプロジェクトを選択します。右クリックして [Execute Project] を選択します。選択したプロジェクトが実行されます。

Execution Monitor が表示されます。詳細については、「Execution Monitor」(88 ページ) を参照してください。

プロジェクトのスケジューリング

プロジェクトをスケジューリングするには、[Tools] - [Runtime Manager] を選択します。Runtime Manager を使用して、タスクの作成、編集、削除、実行、終了を行います。

詳細については、「ジョブとスケジュール・タスクの管理」(66 ページ) を参照してください。

ジョブの管理

ジョブを使用すると、1 つまたは複数のプロジェクトに強力な制御フローを設定できます。ユーザの操作なしで Sybase ETL ジョブを実行するようにスケジュールすることができます。

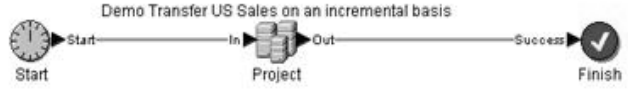
ジョブ内のプロジェクトの成否に応じて、ジョブ実行を制御することができます。

Job コンポーネント

1 つのプロジェクトを実行するジョブは、少なくとも次のコンポーネントで構成されています。

- Start コンポーネント
- Project コンポーネント
- Finish コンポーネント

図 3-2： 最小限のコンポーネントを持つジョブ



ジョブを拡張して次のものを複数含めることができます。

- 連続または並列順のプロジェクト
- シンクロナイザ
- Finish コンポーネントと Error コンポーネント

Start コンポーネントには、常に 1 つ以上の Project コンポーネントが続きます。

図 3-3： 複数のコンポーネントを持つジョブ

❖ **ジョブの作成**

1　ナビゲータでジョブを右クリックし、[New] - [Job] を選択します。使用可能な Job コンポーネントが、コンポーネント・ストアに表示されます。

2　Start コンポーネントをコンポーネント・ストアから設計ウィンドウに追加します。

3　Project コンポーネントを追加し、Start コンポーネントに接続します。

4　Finish コンポーネントを追加し、Project コンポーネントに接続します。

5　Project コンポーネントをダブルクリックします。

6　このジョブに含めるプロジェクトを選択します。[Save] をクリックします。

これでジョブを Sybase ETL Development で実行したり、スケジュール・タスクとして実行したりする準備ができました。

ナビゲータで、ジョブに含まれるプロジェクトを表示してアクセスできます。

❖ **ジョブの変更**

1　ナビゲータでジョブ名をダブルクリックするか、ジョブを右クリックして [Open] を選択します。

2　ジョブを変更し、変更内容を保存します。

❖ **ジョブのコピー**

1　コピーするジョブをダブルクリックして、設計ウィンドウで開きます。

2　ナビゲータでジョブを右クリックし、[Save As] を選択します。または、[File] メニューの [Save As] を選択します。

複数のリポジトリで作業している場合は、ターゲット・リポジトリを選択します。

3 ジョブの名前を指定します。既存のジョブのコピーが作成されます。元のジョブはそのまま残り、元のジョブへの参照も保存されません。

注意 別のリポジトリにジョブをコピーしても、ジョブに含まれるプロジェクトはコピーされません。ジョブ内のすべてのプロジェクトおよびマルチプロジェクト・コンポーネントについて、新しいリポジトリからプロジェクトを選択する必要があります。

❖ ジョブの転送

複数のリポジトリを使用している場合、あるリポジトリから別のリポジトリにジョブおよび含まれているプロジェクトをすべてコピーして、元のオブジェクトへの参照を保持できます。たとえば、開発リポジトリのジョブをテスト・リポジトリまたは運用リポジトリに移動する場合があります。元のジョブへの参照を保存することにより、転送の次回開始時にジョブが認識されて、受信オブジェクトに関連するすべてのものが選択的に置き換えられます。

1 ナビゲータで転送するジョブを右クリックし、[Transfer] を選択します。または、[File] メニューの [Transfer] を選択します。

2 ジョブを転送するリポジトリを選択します。

ジョブおよび含まれているすべてのプロジェクトは、あるリポジトリから別のリポジトリにコピーされますが、元のオブジェクトが参照されます。

注意 転送によってコピーされるのはジョブ定義のみです。パラメータ・セットや実行プロパティなどの関連データは転送されません。

❖ ジョブの削除

1 ナビゲータでジョブを右クリックし、[Delete] を選択します。

2 [Delete] をクリックします。デフォルトでは、選択したジョブのみが削除されます。ジョブおよび含まれるすべてのプロジェクトを

削除するには、[Delete Included Projects] オプションを選択します。

注意 ジョブで使用したプロジェクトを削除する前に、そのプロジェクトが他のジョブで使用されていないことを確認してください。これは自動的には確認されません。設計用に現在開かれており、いずれかのユーザによってロックされているプロジェクトは影響を受けません。

❖ **ジョブ名の変更**

1　ナビゲータでジョブを右クリックします。

2　[Rename] を選択します。

ジョブ実行の制御

ジョブの実行は、次の方法で制御できます。

- Synchronizer コンポーネントを使用します。このコンポーネントを使用すると、プロジェクトの成否に基づいてジョブの実行を分岐することができます。
- 各プロジェクトのエラーを無視します。

詳細については、「Job コンポーネント」(218 ページ) を参照してください。

ジョブの実行

ジョブは、Sybase ETL Development から直接実行することも、オペレーティング・システムのタスク・マネージャでスケジュール・タスクとして特定の時間間隔で実行することもできます。

- 設計ウィンドウで現在開かれているジョブを実行するには、[Run] - [Execute] を選択します。
- ナビゲータからジョブを直接実行するには、ジョブを右クリックし、[Job Execute] を選択します。
- ジョブをスケジューリングするには、[Tools] - [Runtime Manager] を選択します。「ジョブとスケジュール・タスクの管理」(66 ページ) を参照してください。

ジョブのスケジューリング

ジョブをスケジューリングするには、[Tools] - [Runtime Manager] を選択します。Runtime Manager を使用して、タスクの作成、編集、削除、実行、終了を行います。Runtime Manager は Windows のタスク・スケジューリング・マネージャをベースとしているため、現在システム上で定義されている、スケジューリングされたすべての ETL タスクを検索できます。

詳細については、「ジョブとスケジュール・タスクの管理」(66 ページ) を参照してください。

テンプレートを使用したプロジェクトとジョブの作成

テンプレートを使用して、プロジェクトとジョブを自動的に作成できます。

テンプレート・アシスタントを使用したマイグレーション・テンプレートの作成

テンプレート・アシスタントを使用すると、新しいテンプレートを作成するか、既存のテンプレートを使用して、データベース間でデータをマイグレートすることができます。

❖ **マイグレーション・テンプレートの作成**

1 [File] - [New] - [Template] を選択します。または、ナビゲータでテンプレートを右クリックし、このオプションを選択します。

2 マイグレーションの詳細を入力します。

- テンプレートの名前を指定します。この名前はテンプレート・オブジェクトに使用され、生成された変換オブジェクトの修飾名としても使用されます。

- マイグレーションの種類を指定します。

  デフォルトで使用可能なマイグレーションの種類は、*DB to IQ* です。

- [Allow execution on multiple engines] オプションを選択して、複数のエンジンを実行できるようにします。

- [Use IQ Multiplex] オプションを選択して、データを IQ にロードする複数のライタを使用することによってマルチプレックス実行をサポートします。複数のテーブルが IQ データベースにマイグレートされる場合、このオプションを選択します。

  注意 マルチプレックス実行をサポートするには、ETL Development および ETL サーバと同じマシンに Adaptive Server® Anywhere (ASA) 11 ODBC ドライバをインストールする必要があります。

- [Use IQ Client Side Load] オプションを選択して、バルク・ロード・データをリモート・ホスト・マシンにあるファイルから IQ データベースにロードできるようにします。

- [Use IQ Lock Table] オプションを選択して、ターゲット・テーブルを排他モードでロックし、同時実行トランザクションによって更新されないようにします。このオプションを選択すると、他のトランザクションがクエリを実行したり、ロックされたテーブルに対して更新を実行したりできなくなります。[Use IQ Lock Table] オプションはまた、Sybase IQ の同じテーブルをロードする複数のプロジェクトをキューします。

  このオプションを選択する場合、プロジェクトがロックを取得するまでの待機時間である最長ブロック時間を指定する必要があります。

- [Next] をクリックします。

3 ソース・データベースの接続情報を入力し、転送するテーブルを選択します。指定するデータベース接続パラメータについては、「データベース接続の設定」(105 ページ) を参照してください。

注意 データベース接続プロパティは、DB コンポーネントのプロパティと同じです。

指定したデータベースで使用可能なテーブルのリストを表示するには、[Logon] をクリックします。デフォルトでは、各テーブルが転送用に選択されています。転送しないテーブルの選択を解除します。また、1 行以上のテーブル・ローを選択してから、右クリックし、[Exclude] を選択します。テーブルを転送対象に含めるには、右クリックして [Transfer] を選択します。

または、次をクリックすることもできます。

- すべてのテーブルを除外する場合は、[Exclude all objects from transfer] アイコン
- すべてのテーブルを転送対象に含める場合は、[Include all objects in transfer] アイコン

テーブルの詳細情報を表示するには、テーブル・ローを選択して右クリックし、次を選択します。

- [Browse] − テーブル・データを表示します。
- [Count] − 選択したテーブルのレコード数を表示します。すべてのテーブルのレコード数を表示するには、[Count All] をクリックします。

[Next] をクリックします。

4 送信先データベースのデータベース接続プロパティを入力します。指定する必要のあるデータベース接続パラメータの詳細については、「データベース接続の設定」(105 ページ) を参照してください。

使用可能なテーブルのリストを表示するには、[Logon] をクリックします。選択したテーブルのテーブル・データまたはレコード数を表示するには、右クリックして [Browse] または [Count] を選択します。すべてのテーブルのレコード数を表示するには、[Count All] をクリックします。

[Next] をクリックします。

5 転送するテーブルの転送設定を指定します。

a 送信元テーブルのスキーマまたは所有者情報を保持するには、[Preserve schema/owner] オプションを選択します。

注意 送信先データベースに同じスキーマまたは所有者が存在している必要があります。

b ステージのプロパティを入力します。

[Stage] フィールドと [Stage Server] フィールドでは、DB Bulk Load IQ コンポーネントの Load Stage プロパティへのパスを指定します。[Use Pipes] オプションが選択されている場合、パスは自動的に設定されます。[Use Pipes] オプションが選択されていない場合は、パス・デリミタで終わる値を手動で入力します。たとえば、*C:¥ETLStage¥* のように指定します。

これらのプロパティの詳細については、「DB Bulk Load Sybase IQ プロパティ・リスト」(197 ページ) を参照してください。

注意 手順 2 のマイグレーションの詳細ウィンドウにある [Use IQ Client Side Load] オプションを選択すると、[Use Pipes] オプションと [Stage server] フィールドが無効になります。

c　ソース属性を選択します。

デフォルトでは、テーブルのすべての属性が転送対象として選択されます。属性の選択を変更するには、[Columns] フィールドのアイコンをクリックします。

[Select Attribute] ウィンドウで、転送から除外する属性の選択を解除します。1 つまたは複数の属性ローを選択し、右クリックして [Exclude] を選択することで、属性の選択を解除することもできます。

d　送信先テーブルを選択します。

送信元テーブルと送信先テーブルの名前は同じであることが前提となります。別の名前を使用するには、[Destination] フィールドに新しい名前を入力するか、既存のテーブルを選択します。

e　各テーブルに適したアクションを実行するための追加オプションを指定します。

- データ・モデル・オプション － 転送が開始される前に、送信先テーブルが存在することを確認します。データ・モデル・オプションは、送信先データ・モデルの設定に役立ちます。これらのオプションは実行には影響を与えませんが、テンプレートから作成したデータ・モデルには影響を与えます。

  選択したソース属性に基づいて、既存でない送信先テーブルを作成するには、[Create Table] オプションを選択するか、オプションを右クリックして [Activate] を選択します。既存のテーブルを再作成するには、[Drop Table] オプションを選択します。

- 実行オプション − これらのオプションは、プロジェクト・レベルの実行に影響を及ぼします。

  ロード前に送信先テーブルのすべてのレコードを削除するには、[Truncate] オプションを選択します。このオプションは、ターゲット・コンポーネントの Truncate Table プロパティに対応しています。

  重大なプロジェクトでエラーが発生すると、ジョブの実行が中断され、エラー信号が送信されます。[Critical] オプションと [Ignore Errors] オプションは、マルチプロジェクト・ジョブ・コンポーネントのプロパティに対応しています。

  [Ignore Errors] 設定は、このテンプレートを使用して生成されたプロジェクトには影響を与えません。

6 収集したデータに対して実行するタスクを選択します。

注意 ここで説明するタスクはすべて、テンプレートの保存を除き、ナビゲータで保存されているテンプレートを右クリックして実行することもできます。

- [Save template] − このオプションを選択すると、テンプレートがリポジトリに保存されます。テンプレートを保存すると、収集したデータを同様のジョブで再利用できます。
- [Build projects and jobs] − 送信元テーブルごとに 1 つのプロジェクトと、すべてのプロジェクトの実行を制御するマイグレーション・ジョブを作成するには、このオプションを選択します。
- [Create the destination data model] − 入力したデータ・モデル・オプションに従って送信先データ・モデルを設定するには、このオプションを選択します。送信先テーブルの作成前に実行される SQL コマンドを入力するには、[Advanced] をクリックします。
- [Execute job] − [Execute job] オプションは、[Build projects and jobs] オプションが選択されている場合にのみアクティブになります。このオプションを選択すると、マイグレーション・テンプレートのデータが処理された後に、生成されたジョブが実行されます。

[Finish] をクリックします。

注意 収集したデータが失われないようにする場合は、少なくとも [Save template] オプションまたは [Build Projects and jobs] オプションを選択するようにしてください。

注意 生成されたジョブを実行する前に、エンジンを登録するか、ジョブを開いて [Multi Engine Execution] オプションを非アクティブにしてください。「複数のエンジンの使用によるジョブ実行時間の短縮」(85 ページ) を参照してください。

データの処理中に、現在の状態や進行状況を表示することができます。

マイグレーション・テンプレートの管理

❖ **テンプレートの作成**

1 ナビゲータでテンプレートを右クリックします。

2 [New] - [Template] を選択します。

テンプレート・アシスタントの指示に従って、マイグレーション・テンプレートを設定します。

❖ **テンプレートの変更**

1 ナビゲータでテンプレートをダブルクリックするか、変更するテンプレートを右クリックして [Open] を選択します。

2 テンプレートに変更を加えて保存します。

❖ **テンプレートのコピー**

1 ナビゲータでテンプレートを右クリックします。

2 [Copy] を選択します。新しいテンプレートの名前を入力します。テンプレートを別のリポジトリにコピーすることもできます。

❖ **テンプレートの削除**

1 ナビゲータでテンプレートを右クリックします。

2 [Delete] を選択します。

注意 テンプレートを削除しても、そのテンプレートを基にしているジョブおよびプロジェクトは影響を受けません。

❖ **テンプレート名の変更**

1 ナビゲータでテンプレートを右クリックします。

2 [Rename] を選択し、テンプレートの新しい名前を入力します。

❖ **テンプレートからのジョブの作成**

保存されているテンプレートに基づいてマイグレーション・ジョブとすべての関連プロジェクトを作成するには、次の手順に従います。

1 ナビゲータでテンプレートを右クリックします。

2 [Build] を選択します。一意の名前を徹底するために、作成タイムスタンプがすべてのオブジェクト名に追加されます。

❖ **テンプレートからのデータ・モデルの作成**

テンプレートに保存されたデータ・モデル・オプションに従って送信先データ・モデルを設定するには、次の手順に従います。

1 ナビゲータでテンプレートを右クリックします。

2 [Create Data Model] を選択します。

サンプル・プロジェクトの作成とシミュレーション

この項では、コンポーネントを使用してサンプル・プロジェクトを作成し、シミュレートする方法について説明します。コンポーネント、プロパティ、および機能の詳細については、第 5 章 コンポーネントを参照してください。

プロジェクトには通常、次の 1 つまたは複数の要素が含まれています。

- プロジェクト・データ・ストリームにデータ・フィードを提供するデータ・プロバイダ

- フィールド値を変換または再マップするデータ・トランスフォーマ
- 変換された値をターゲットに書き込むデータ・シンク

注意 この項の結果は、デフォルト・リポジトリの Demo Getting Started プロジェクトで確認できます。

データ・プロバイダの追加

次のいずれかの方法で、DB Data Provider Full Load をプロジェクトに追加します。

- コンポーネント・ストアの [Source] タブから設計ウィンドウに、コンポーネントをドラッグします。
- コンポーネント・ストアで追加するコンポーネントを右クリックし、[Add] を選択します。
- コンポーネント・ストアで追加するコンポーネントをダブルクリックします。

コンポーネントを設計ウィンドウに追加すると、コンポーネントのデフォルト設定が表示されます。

注意 設定ウィンドウに**太字**で表示されているプロパティは必須です。

❖ データ・プロバイダの設定

1 [Interface] メニューの [ODBC] を選択します。さまざまなインタフェースの種類については、「データベース接続の設定」(105 ページ) を参照してください。

2 [Host Name] メニューで [ETLDEMO_US] を選択します。

コンポーネントの初期設定を確認すると、設定がプロパティ・ウィンドウに表示されます。

3 データ・ソースから取得する情報を定義するには、[Query] フィールドの [Query] アイコンをクリックします。

4 SQL クエリを入力するか、[Query Designer] アイコンをクリックして必要な SQL を生成します。

[Query Designer] ウィンドウの左側のウィンドウ枠で、接続されているデータベースのテーブル・カタログをナビゲーションできます。

5 1 つまたは複数のテーブルを追加するには、テーブル名を設計ウィンドウにドラッグするか、テーブル名を右クリックして [Add Object to Query] を選択します。

6 PRODUCTS テーブルをクリックして、設計ウィンドウにドラッグします。

7 [Save] をクリックして Query Designer を閉じ、[Query] ウィンドウに戻ります。select クエリが自動的に生成されます。

8 [Execute the Query] アイコンをクリックして、クエリを実行またはテストします。また、クエリを変更することもできます。

9 [Save] をクリックして [Query] ウィンドウを閉じます。

注意 コンポーネントの設定が正常に完了すると、関連するポートの色が赤色または黄色から緑色に変わります。

データ・シンクの追加

次のいずれかの方法で、DB Data Sink Insert をプロジェクトに追加します。

- コンポーネント・ストアの [Destination] タブから設計ウィンドウに、コンポーネントをドラッグします。
- コンポーネント・ストアで追加するコンポーネントを右クリックし、[Add] を選択します。
- コンポーネント・ストアで追加するコンポーネントをダブルクリックします。

コンポーネントを設計ウィンドウに追加するとすぐに、コンポーネントのデフォルト設定が表示されます。

注意 設定ウィンドウに**太字**で表示されているプロパティは必須です。

❖ データ・シンクの設定

1 [Interface] メニューの [ODBC] を選択します。さまざまなインタフェースの種類については、「データベース接続の設定」(105 ページ) を参照してください。

2 [Host Name] メニューで [ETLDEMO_DWH] を選択します。

3　[Destination Table] フィールドに PRODUCTS と入力します。または、[Destination table] アイコンをクリックし、表示されるリストで [PRODUCTS] を選択します。

4　[Finish] をクリックして、設定を確定します。

これでプロジェクトに 2 つのコンポーネントができました。[File] - [Preferences] ウィンドウで [Create automatic links when components are added] オプションを選択した場合、コンポーネント間のリンクが自動的に作成されます。リンクが自動的に作成されない場合は、出力ポートをクリックしてデータ・シンクの入力ポートにドラッグすると作成されます。

DB Data Provider Full Load コンポーネントの出力ポート (OUT ポート) と DB Data Sink Insert コンポーネントの入力ポート (IN ポート) が両方とも緑色で表示されます。これは両方のコンポーネントが設定されたことを示します。

DB Data Sink Insert コンポーネントのプロパティ・ウィンドウで、選択したコンポーネントのすべてのプロパティを確認したり設定したりできます。

❖ 属性マッピングの確認と定義

1　コンポーネント間のリンクを右クリックします。リンクの色が緑色に変わります。

2　[Mapping] を選択します。

データ・ソースとターゲット・ソース間のマッピングが自動的に作成されます。マッピングを変更するには、接続している線を選択し、別の接続ポイントに付加します。

注意 割り当てられていないターゲット接続ポイントにのみマッピングできます。すべてのターゲット接続ポイントが割り当て済みの場合は、現在リンクされているマッピング・ラインを選択して削除することによって、接続ポイントの割り当てを解除してください。削除するには、マッピング・ラインを選択して [Delete] キーを押すか、右クリックして [Delete] を選択します。

Data Calculator の追加

1 コンポーネント・ストアの [Transform] タブをクリックします。

2 Data Calculator Javascript コンポーネントを選択し、既存のコンポーネントを接続しているリンクにドロップします。リンクの色が青色に変わります。

Data Calculator コンポーネントを解放すると、次のようになります。

- Data Calculator コンポーネントが左右のコンポーネントにリンクされます。
- [Data Calculator] ウィンドウが表示されます。

[Data Calculator] ウィンドウには、テーブル・ビューとグラフ・ビューがあります。

- テーブル・ビューは、変換規則を入力するために使用します。
- グラフ・ビューは、入力ポートと出力ポート間のマッピング・シーケンスを視覚的に定義するために使用します。

3 [Graph] タブをクリックします。2 つのセクション － IN と OUT は、ポート属性の現在の構造を表します。

4 デフォルトのマッピングを順番に割り当てる場合は、[Yes] をクリックします。

5 [Tabular] タブをクリックして、テーブル・ビューに戻ります。

6 PR_NAME 属性のすべての受信データを大文字に変更します。

```
uUpper(IN.PR_NAME) ' OUT.PR_NAME
```

7 IN.PR_NAME 属性の [Transformation Rule] カラムに uUpper(IN.PR_NAME) と入力します。関数を追加しなくても、IN.PR_NAME 値は OUT.PR_NAME 属性に転送されます。

8 [Save] をクリックして、設定を確定します。プロジェクトのすべてのポートが緑色で表示されます。これは、すべてのコンポーネントが正常に設定されたことを示します。

9 [File] - [Save] を選択します。

シミュレーションの開始

1　ツールバーの [Start] アイコンをクリックして、すべてのコンポーネントを初期化します。

2　コンポーネントごとにプロジェクトをステップ実行するには、[Step] をクリックします。

シミュレーションの任意の時点で、現在のデータ・セットをプレビューできます。たとえば、最初のステップが実行されると、データ・レコードは Source コンポーネントから Data Calculator に転送されます。リンク上の数値は、転送されたレコードの数を示します。

3　リンクを右クリックし、[Preview] を選択すると、リンク上のデータをプレビューできます。

注意 処理レコードがない場合や、使用可能なシミュレーション・データがない場合、[Preview] オプションは無効になります。

第 4 章

高度な概念とツール

Query Designer

Query Designer は次の目的に使用します。

- 現在のプロジェクトの接続されているデータベースのテーブル・カタログを参照する。
- グラフィカル・ユーザ・インタフェースを使用して SQL クエリを作成する。
- 生成された SQL 文を確認する。
- データベースに対して SQL クエリを実行する。
- 選択したテーブルまたはビューでデータを参照する。
- スキーマにテーブルを作成する。

- テーブルのすべてのレコードを削除する。

注意 テーブルのすべてのレコードを削除するには、[File] - [Preference] ウィンドウの [Enable delete functionality of database objects] オプションを選択します。詳細については、「設定のカスタマイズ」(22 ページ) を参照してください。

- テーブルまたはビューのレコード数をカウントする。

Query Designer を開く

この項では、デモ・リポジトリの Demo Getting Started プロジェクトを使用して、Query Designer でクエリを作成します。

Query Designer を開くには、次の手順に従います。

1 ナビゲータで Demo Getting Started プロジェクトをダブルクリックして、設計ウィンドウで開きます。

2 設計ウィンドウで DB Data Provider Full Load コンポーネントをダブルクリックします。または、コンポーネントを選択して、そのプロパティをプロパティ・ウィンドウに表示します。

3 [Query] アイコンをクリックします。

4 [Query Designer] アイコンをクリックします。

Query Designer のインタフェース

Query Designer のインタフェースは、次の要素で構成されています。

- [Query Definition] ウィンドウ枠 – 作業対象のレコードに固有の select 文を自動的に生成するために使用する設計ウィンドウが含まれます。
- ナビゲータ – すべてのテーブルとビューが [Model] タブに表示され、最近使用したテーブルまたはビューが [Recent] タブに表示されます。[File] - [Preferences] ウィンドウの [Recent] タブに表示されるテーブルまたはビューのデフォルト数を設定できます。詳細については、「設定のカスタマイズ」(22 ページ) を参照してください。
- 属性タブ ([Select]、[Join]、[Where]、[Sort]、[Group]) と [Generated Query] タブ – 属性の詳細に加えて、生成されたクエリをクエリ作成中に表示できます。

クエリの作成

この項では、デモ・リポジトリの Demo Getting Started プロジェクトを使用してクエリを作成します。

❖ 簡単なクエリの作成

テーブルからすべての属性を取得する簡単なクエリを生成するには、PRODUCTS テーブルを使用します。

1　ナビゲータで [Model] タブを選択し、テーブルまたはビューの名前をクリックします。特定のテーブル名またはビュー名を検索するには、[CTRL + F] キーを押します。

2　選択したオブジェクトを設計ウィンドウにドラッグします。

3　生成されたクエリの結果を表示するには、[View] - [Generated Query] を選択するか、[Generated Query] タブを選択します。

❖ 複数のテーブルを使用したクエリの作成

2 つのテーブルをジョインして情報を取得するクエリを生成するには、PRODUCTS テーブルと SALES テーブルを使用します。

1　PRODUCTS テーブルをナビゲータから設計ウィンドウにドラッグします。

2　SALES テーブルをナビゲータから設計ウィンドウにドラッグします。

3　両方のテーブルの PR_ID フィールドをリンクして、テーブル間にジョインを作成します。Query Designer で複数のテーブルまたはビュー内の同じ名前の属性間にジョインを自動的に作成するには、次の手順に従います。

   a　Sybase ETL Development のメイン・ウィンドウから [File] - [Preferences] を選択します。

   b　[Workbench] - [Query Designer] を選択し、[Create joins automatically] オプションを選択します。詳細については、「設定のカスタマイズ」(22 ページ) を参照してください。

4　ジョイン属性の詳細を表示するには、Query Designer ウィンドウの [Join] タブをクリックします。

❖ **ジョインのデフォルト設定の変更**

2 つのテーブル間のジョインは、ジョイン・フィールドを接続する線によって示されます。この線には、ジョイン演算子のラベルが付けられています。デフォルトでは、等価ジョインです。

1 2 つのジョイン・フィールドを接続する線を右クリックします。

2 [Modify] を選択します。

3 ジョインのタイプを選択します。

❖ **ジョインのソート順の変更**

1 [Join] タブでローを右クリックし、次のいずれかを選択します。

- [Move to start]
- [Move up]
- [Move down]
- [Move to end]

2 ジョインのデフォルト状態に戻すには、ローを右クリックし、[Sort joins to default order] を選択します。

❖ **select 句への 1 つまたは複数の属性の追加**

1 PRODUCTS テーブルと SALES テーブルが設計ウィンドウにない場合は、これらのテーブルを設計ウィンドウにドラッグします。

2 1 つの属性を追加するには、追加する属性を右クリックし、[Add Items to Selection] を選択します。複数の属性を追加するには、[Ctrl] キーを押しながら、追加する属性を選択します。

または、[Query Definition] ウィンドウ枠の [PRODUCTS] および [SALES] タブをクリックし、select 句に追加する属性を選択します。属性を検索するには、[Search] アイコンをクリックし、検索条件を入力します。

注意 アスタリスク (*) をワイルドカードとして使用すると、任意の数の不明な文字を検索できます。

次に例を示します。

- 属性が integer データ型の場合、検索条件を次のように指定できます。

  ```
  int、int*、i*ger
  ```

- 属性名に “PROD” が含まれ、その末尾が “CD” の場合、検索条件を次のように指定できます。

  ```
  *PROD*CD
  ```

❖ **選択したテーブルのすべての属性を select 句に追加**

1 [Query] タブでテーブルのヘッダをクリックします。

2 右クリックし、[Add Items to Select] を選択します。

❖ **属性の詳細と生成されたクエリの表示**

1 Query Designer によって生成されたクエリを表示するには、[View] - [Generated Query] を選択するか、[Generated Query] タブを選択します。

2 適切なタブをクリックして、属性の詳細を表示します。

❖ **select 属性への関数の追加**

1 [Select] タブで、関数を追加する属性を右クリックします。

2 追加する関数を選択します。

Content Explorer

Content Explorer を使用して、接続されているすべてのデータ・ソースのスキーマ情報とデータ内容をブラウズします。ファイルやリポジトリに保存できないアドホック・クエリを生成する場合にのみ Content Explorer を使用します。生成された SQL を保存するには、生成されたクエリを [Generated Query] ウィンドウで選択してコピーします。[Generated Query] ウィンドウを開くには、[View] メニューの [Generated Query] を選択します。また、[Generated Query] タブでクエリを選択してコピーすることもできます。

Content Explorer を開くには、次のいずれかの方法を使用します。

- [Tools] - [Content Explorer] を選択します。データ・ソースに現在接続されているすべてのコンポーネントが [Choose Data Source] ウィンドウに表示されます。現在接続されているデータベースのリストに表示される名前は、ユーザ定義の名前とコンポーネント・タイプの汎用名の組み合わせです。コンポーネントを選択し、[Start] をクリックして Content Explorer を開きます。
- データベース・コンポーネントを右クリックし、[Content Explorer] を選択します。

File Log Inspector

プロジェクトおよびジョブの実行や致命的なエラーに関する情報を検査したり、システム・ログを確認したりするには、File Log Inspector を使用します。

1 [Tools] - [File Log Inspector] を選択します。生成されたログ・ファイルが表示されます。

2 表示するログ・ファイル情報をクリックします。次のいずれかまたはすべてのログ・ファイルを表示できます。

注意 ログ・ファイルは、インストール・ディレクトリの *¥log* サブディレクトリにあります。

- *execution.log* － ジョブとプロジェクトの実行に関する情報が記録されます。
- *fatal.log* － 予期しない深刻な動作が発生した場合に書き込まれる低レベルの情報が記録されます。この内容には、システムが *system* ログ・ファイルに書き込むことができなくなったときの致命的なシステム例外の情報が含まれます。
- *system.log* － システム・アクティビティに関する情報に加えて、操作イベントと例外イベントに関する情報が記録されます。*system.log* ファイルのエラー・コードをチェックすると、Sybase ETL の操作中に発生したエラーの原因と、考えられる解決方法を判別できます。次の表は、*system.log* ファイルで発生する可能性があるエラー・コードを示します。

エラー・コード	種類	説明
0	情報	ジョブまたはプロジェクトが正常に実行されました。
100 または 110	エラー	ETL エンジンの初期化のエラー。
101	エラー	無効なライセンスのエラー。
1100	エラー	ETL 例外障害。正しくないコマンド・ラインの使用を含んでいます。
1103 または 1104	エラー	指定されていない例外による障害。
10001	エラー	ジョブ、プロジェクト、パラメータ・セットを含むリポジトリから情報を取得できませんでした。
10005	エラー	ジョブの実行ができませんでした。
10006	エラー	プロジェクトの実行ができませんでした。
10101	エラー	リポジトリ・データベースへの接続に失敗しました。

system.log ファイルに書き込まれるデータの詳細レベルは、設定されているトレース・レベルに応じて異なります。

トレース・レベルを設定するには、次の手順に従います。

- [Tools] - [Enable System Trace] を選択します。
- JavaScript プロシージャの uTracelevel(*n*) 関数を使用します。

  uTracelevel(*n*) 関数 (*n* は 0 ～ 5 の値) を使用すると、コンポーネントの内部からトレース・レベルを設定できます。
- インストール・フォルダの *etc* サブディレクトリにある *default.ini* ファイルで次の行を変更して、トレース・レベルを指定します。

  ```
  Tracelevel=value
  ```

  ここで、*value* は、設定するトレース・レベルです。変更内容を有効にするには、Sybase ETL を再起動する必要があります。

注意 トレース・レベルは 0 または 5 のいずれかに設定することをおすすめします。

3 (オプション) − ログをトランケートするには、ツールバーの [Truncate log] アイコンをクリックします。[Yes] をクリックして操作を確定します。

4 (オプション) − 特定のログ・ファイルを検索するには、ツールバーの [Select rows containing a string or regular expression] アイコンをクリックします。

注意 Log File Inspector では、ログ・ファイルの最新の 1 MB のみ表示されます。サイズが大きいログ・ファイルでそれ以前のレコードを表示するには、そのファイルをテキスト・エディタで開きます。

ジョブとスケジュール・タスクの管理

プロジェクトとジョブを管理し、現在のスケジュール・タスクの概要を確認するには、Runtime Manager を使用します。タスクを作成、編集、削除、実行、終了するには、Runtime Manager のスケジューリング・ウィザードを使用します。

Runtime Manager は Windows のタスク・スケジューリング・マネージャをベースとしているため、現在システム上で定義されている、スケジューリングされたすべての ETL タスクが含まれます。

❖ 新しいスケジュールの作成

1 [Tools] - [Runtime Manager] を選択します。

2 [Actions] - [Create] を選択します。または、ツールバーの [Create a new schedule] アイコンをクリックします。

3 実行するプロジェクトまたはジョブを選択します。[Next] をクリックします。

4 スケジュール名を入力します。開始時刻と終了日を入力し、タスクの実行頻度を指定します。

- [Daily] − 毎日、指定した時刻にタスクを実行します。
- [Weekly] − 毎週、特定の曜日の指定した時刻にタスクを実行します。曜日を指定します。
- [Monthly] − 選択した各月で特定の日の特定の時刻にタスクを実行します。月の日付を指定し、適切な月を選択します。

- [Once] − 指定した日付と時刻に一度だけタスクを実行します。
- [At Login] − ログイン時にタスクを実行します。
- [At System Startup] − システム起動時にタスクを実行します。

[Next] をクリックします。

5　スケジュールを実行する Windows ユーザ・アカウントのユーザ名とパスワードを入力します。パスワードを確認します。[Next] をクリックします。

注意 有効な Windows ユーザ・アカウントを入力する必要があります。インストールの種類に応じて、ユーザは ETL ユーザ・フォルダ (インストール・ディレクトリなど) または Windows ユーザ・ディレクトリに対する読み取りまたは書き込みアクセス権を持っている必要があります。

6　スケジュールが正常に作成されると、メッセージが表示されます。[Finish] をクリックします。

新しいスケジュールが Runtime Manager に表示されます。既存のスケジュールが存在する場合は、既存のスケジュールも一緒に表示されます。

❖ スケジュールの編集

1　編集するスケジュール・プロジェクトまたはジョブを選択します。

2　ツールバーの [Edit a Schedule] アイコンをクリックするか、[Actions] メニューから [Edit] を選択します。

❖ スケジュールの実行

1　実行するスケジュール・プロジェクトまたはジョブを選択します。

2　ツールバーの [Execute a Schedule] アイコンをクリックするか、[Actions] メニューから [Execute] を選択します。

❖ スケジュールの削除

1　削除するスケジュール・プロジェクトまたはジョブを選択します。

2　ツールバーの [Delete a Schedule] アイコンをクリックするか、[Actions] メニューから [Delete] を選択します。

❖ **スケジュールの終了**

1 終了するスケジュール・プロジェクトまたはジョブを選択します。

2 ツールバーの [Terminate a running Schedule] アイコンをクリックするか、[Actions] メニューから [Terminate] を選択します。

スケジュールしたプロジェクトまたはジョブは、Windows のタスク スケジューラの制御下で実行されます。

タスク スケジューラのジョブ実行ステータス・コードを次の表に示します。

表 4-1：ジョブ実行ステータス・コード

結果	種類	説明
0	情報	ジョブまたはプロジェクトが正常に実行されました。
1	警告	ジョブまたはプロジェクトの実行がキャンセルされました。
101	警告	有効なライセンスがありません。
10001	エラー	リポジトリからジョブ・データを取得できません。
10002	エラー	ジョブの初期化コンポーネントが見つかりません。
10003	エラー	外部ジョブ設定を初期化できません。
10004	エラー	初期化に失敗しました。
10005	エラー	ジョブまたはプロジェクトの実行に失敗しました。
10101	エラー	リポジトリ・データベースへの接続に失敗しました。
10102	エラー	接続されているデータベースで有効なリポジトリが見つかりませんでした。
10103	エラー	接続されているリポジトリでセッションを開けません。

SQL のカスタマイズと変換規則

プロジェクトまたはジョブを設定するときに、次の内容をカスタマイズできます。

- Source コンポーネントの設定用の SQL クエリ
- タスクの前処理および後処理用の SQL コマンド
- 変換プロセスを操作するための式、条件、プロシージャ

SQL コマンドのフォーマットは、コンポーネントに接続されているデータベース・システムに応じて異なります。ただし、Sybase ETL (JavaScript) でサポートされている変換言語のフォーマットは、プロジェクトで使用するソースまたはターゲット・システムに関係なく同じです。

JavaScript 式を角カッコ表記 (SBN) 式に含めることができます。この処理を行うと、カスタマイズの労力が大幅に削減されます。SBN 式は、コンポーネント・プロパティ (特定のプロパティに対して Evaluate オプションがアクティブな場合)、SQL 文、または任意の前処理または後処理コマンドの一部として使用できます。SBN 式は、開始および終了角カッコの間に配置します。設計時に定義する定数式とは異なり、SBN 式は実行時に評価されます。

式とプロシージャ

式は、識別子と 1 つの値を計算できる演算子の組み合わせです。単純式は、変数、定数、属性、またはスカラ関数です。演算子を使用して、2 つ以上の単純式を複合式にジョインできます。

式の例を次に示します。

```
'Miller'
uConcat("Time ", "goes by")
(uMid(SA_ORDERDATE, 1, 10) >= '1998-01-01')
[uTracelevel(3)]
```

プロシージャは、式、文、および制御構造体を含むプログラミング単位です。たとえば、JavaScript で次のようにプロシージャを記述できます。

```
if (IN.PR_PRICE < 250)
       OUT.PR_GROUP2 = 'low end' ;
else {
    if (IN.PR_PRICE < 1000)
       OUT.PR_GROUP2 =  'mid range';
    else
       OUT.PR_GROUP2 = 'high end';
}
```

変数

変数は、値のシンボリック名です。変数には 2 つの基本プロパティがあります。

- スコープ
- データ型

変数のスコープによって、環境のどのスコープで変数を参照できるかが決定されます。

変数には 2 つの種類があります。

- ポート変数
- コンポーネント変数

ポート変数

ポート構造体の値は、コンポーネント内のポート変数の一部として参照されます。IN ポートと OUT ポートの両方の自動ポート変数があります。ポート変数はコンポーネント内で有効で、ポート構造体の名前とデータ型を継承します。変数の名前には、IN ポートの場合は IN. プレフィクスが付けられ、OUT ポートの場合は OUT. プレフィクスが付けられます。IN ポート変数は読み取り専用ですが、OUT ポート変数は書き込み可能です。

ポート変数を式で使用した例を次に示します。

```
uUpper(IN.CU_NAME)
```

ポート変数をプロシージャで使用した例を次に示します。

```
OUT.CU_NAME = uUpper(IN.CU_NAME);
```

コンポーネント変数

コンポーネント変数は、コンポーネント・プロパティに関連付けられ、現在評価されているプロパティの内容を表します。これらの変数はコンポーネントの内部で参照できます。次に示すように、変数の名前には、REF. プレフィクスが付けられます。

```
uIsNull(REF.Host)
```

変換での柔軟性を高めるために、すべてのポート変数とコンポーネント変数は内部でデータ型 string を使用します。そのため、数値を使用すると予期しない動作が発生することがあります。

1 で乗算された場合、計算では string 型の変数の数値が使用されます。

```
IN.Margin="2", IN.Price="10"
IN.Margin>IN.Price  - returns TRUE
```

数値比較を適用するには、次のようにします。

```
IN.Margin*1>IN.Price*1  - returns
FALSE
```

ポート変数とコンポーネント変数の名前には、次の JavaScript キーワードを使用しないでください。

予約されている JavaScript キーワード

break	case	catch	continue	default
delete	do	else	finally	for
function	if	in	instanceof	new
return	switch	this	throw	try
typeof	var	void	while	with
abstract	boolean	byte	char	class
const	debugger	double	enum	export
extends	final	float	goto	implements
import	int	interface	long	native
package	private	protected	public	short
static	super	synchronized	throws	transient
volatile	const	export		

関数

Sybase ETL には、Unicode 文字セットをサポートする設計に基づいて関数と演算子の完全なセットが含まれています。

Sybase ETL 関数には u プレフィクスが付けられています (uConcat() など)。

角カッコ表記

式と SQL 文には、角カッコ表記 (SBN) 式を含めることができます。SBN 式は、Sybase ETL サーバ で式または SQL 文が実行される前に評価されます。SBN 式は、角カッコ [..] で囲みます。「SBN 式」という表記は、間接式の同意語として使用されます。

SBN 式は以下で使用できます。

- 式
- SQL クエリ
- pre-SQL 文と post-SQL 文
- 変換規則
- ファイル名
- パス定義
- コンポーネント・プロパティ
- URL

例

リテラルは、引用符で囲まれた文字列です。リテラルで SBN を使用した場合、SBN が最初に評価されます。

```
'Arrival Date: [uDate('now', 'localtime')]'
```

次の式は、Text Data Provider でファイルのパスを指定します。

```
[uSystemFolder('APP DEMODATA')]¥PRODUCTS.TXT
```

注意 コンポーネントのプロパティ・ウィンドウの [Eval] カラムは、入力された値が評価されて SBN 式が解決されるかどうかを示します。多くのプロパティ項目では、[Eval] カラムはオプションです。[Eval] チェックボックスを切り替えるには、プロパティ項目を右クリックし、[Evaluate] を選択します。

SQL 文

SQL クエリは、データを抽出するすべてのコンポーネント (主に Data Provider および Staging コンポーネント) に使用します。クエリは OUT ポート構造体を定義するので、これらのコンポーネントで必須です。

コンポーネントのクエリを入力するには、プロパティ・ウィンドウで [Query] アイコンをクリックします。次のことができます。

- クエリの入力。
- クエリの実行。
- クエリの保存。
- Query Designer を開く。詳細については、「Query Designer」(59 ページ) を参照してください。
- データベース・スキーマの検索。

クエリの入力

1　[Query] フィールドに select 文を入力します。Query Designer を使用してアドホック・クエリを作成することもできます。詳細については、「Query Designer」(59 ページ) を参照してください。クエリのテーブルと属性は、使用可能なデータベース・スキーマから選択できます。「データベース・スキーマの検索」を参照してください。

2　[Execute Query] アイコンをクリックします。

注意 既存の select 文を変更した場合は、別の手順を実行する前にコンポーネントを初期化します。

クエリの検証

クエリは、コンポーネントに接続されているデータベース・システムに対して直ちに検証されます。したがって、クエリ構文は、接続されているデータベース・システムで使用されているネイティブの SQL 言語に準拠している必要があります。SQL92 ANSI 標準クエリを使用すると、select 文を変更することなく、別のデータベース・システムに切り替えられます。

データベース・スキーマの検索

クエリのテーブルと属性は、使用可能なデータベース・スキーマから選択できます。

1 プロパティ・ウィンドウで [Query] アイコンをクリックします。

2 [Database Lookup] アイコンをクリックします。

3 クエリで使用するテーブルと属性を選択し、[OK] をクリックします。

クエリでの SBN 式の使用

次の例は、クエリで SBN 式を使用する方法を示しています。

例

特定の顧客レコードを取得する select 文は、そのレコードの定数顧客レコード CU_NO を含むことができます。

```
select * FROM CUSTOMERS WHERE CU_NO = '12345678'
```

SBN を使用すると、CU_NO の定数値をカスタム・コンポーネント・プロパティに割り当てることによって、さらに柔軟なアプローチを使用できます。詳細については、「カスタム・プロパティ」(100 ページ) を参照してください。プロパティ CustNo に値 “12345678” が割り当てられている場合、動的な式を含む select 文は次のようになります。

```
select * FROM CUSTOMERS WHERE CU_NO = '[REF.CustNo]'
```

SBN 式の内部では、任意の Sybase ETL 関数を使用できます。次の文は、CustNo1 に *“1234”* の値を使用し、CustNo2 に *“4567”* の値を使用して、同じレコードを返します。

```
select * FROM CUSTOMERS WHERE CU_NO = '[uConcat
(REF.CustNo1, REF.CustNo2)]'
```

前処理 SQL 文と後処理 SQL 文

データベース・コネクティビティを含むコンポーネントでは、コンポーネントのプロパティ・ウィンドウに前処理 SQL 文と 後処理 SQL 文を入力できます。これらのプロパティを使用すると、コンポーネントの初期化 (前処理) 中に実行される SQL 文、またはプロジェクトの完了後 (後処理) に実行される SQL 文を入力できます。

いくつかの注意事項があります。

- 実行後、SQL 文は出力を返しません。
- 接続されているデータベース・システムで受け入れられる SQL 文を使用できます。

- セミコロンをデリミタとして使用することによって、複数の SQL 文を前処理または後処理 SQL プロパティに入力できます。
- 前処理および後処理 SQL で SBN 式を使用できます。

前処理および後処理 SQL の例を次に示します。

```
delete from products;
update customers
set cu_desc = 'valid';
```

JavaScript Editor and Debugger の使用

JavaScript は、他の製品やアプリケーションに埋め込むために設計されたオブジェクト指向スクリプト言語です。JavaScript では、プログラムによってオブジェクトを制御するためのオブジェクト操作が可能です。

JavaScript の機能は、言語の柔軟性を向上させるグリッド関数によって強化されています。JavaScript Editor and Debugger を使用すると、JavaScript コードを対話的に編集、デバッグ、実行できます。

機能

JavaScript Editor and Debugger は、受信データの変換規則を設定するために主に使用されます (その他の目的でも使用されます)。JavaScript Editor and Debugger の内部では、1 つの入力レコードを使用してスクリプトを実行およびテストできます。

JavaScript Editor and Debugger には、次の機能があります。

- 読みやすいように色分けされた構文
- コードを実行またはステップ実行するときに、変数および属性の値を制御するウォッチリスト
- 任意の行位置でコードの実行を停止する複数のユーザ定義のブレークポイント
- コードの実行開始位置を選択できるユーザ定義の実行ポイント
- 行ごとにコードを実行するステップ・モード
- デバッグ時のステップオーバー
- JavaScript 式の評価
- コード実行の結果の検証

JavaScript Editor and Debugger の起動

1 Data Calculator JavaScript コンポーネントをダブルクリックするか、プロパティ・ウィンドウの [Rule] アイコンをクリックします。

2 [Transformation Rule] カラムでローを選択し、[Edit] アイコンをクリックします。

[JavaScript Editor and Debugger] ウィンドウには、次の内容が表示されます。

- ナビゲータ − [Variables] タブと [JavaScript] タブが含まれます。[Variable] タブには、入出力ポート変数、テンポラリ変数、事前定義変数が表示されます。[JavaScript] タブには、プロシージャ内で適用できるすべての関数、コマンド、システム変数が表示されます。
- [Edit/Debug] ウィンドウ枠 − 実際のコードを編集できます。
- 次のタブは、ウィンドウの下部に表示されます。
  - [Tasks] − プロシージャがコンパイルされた後の検証結果が表示されます。
  - [Watch List] − デバッグ中にコードをステップ実行するとき、選択した変数とその値が表示されます。
  - [Input Records] − 現在の入力レコードの内容が表示されます。入力レコードと出力レコードの同期をとるには、ツールバーの [Start debugging] アイコンをクリックします。
  - [Output Record] − 現在の出力レコードの内容が表示されます。
  - [Expression] − JavaScript 式を入力して、ツールバーの [Evaluate] をクリックすると、式の結果が表示されます。

編集モードとデバッグ・モード

JavaScript Editor and Debugger は、編集モードで起動します。デバッグ・モードに切り替えるには、[Debug] - [Start] を選択します。

デバッグ・モードでは、編集領域の背景が濃い灰色で表示されます。編集モードに切り替えるには、ツールバーの [Stop debugging and start editing] アイコンをクリックします。

JavaScript の編集とデバッグ

JavaScript コードを検証するには、[Debug] - [Start] を選択します。検証の結果は、[Tasks] タブに表示されます。

JavaScript Editor には、スクリプトの実行をトレースする十分な機能があります。行ごとにコードをステップ実行することや、あるブレークポイントから別のブレークポイントまでステップ実行することができます。変数の現在の値は、いつでも確認できます。

注意 コメント行の先頭には、2 つのスラッシュ (//) を付けます。

❖ コードのステップ実行

注意 JavaScript Editor and Debugger は、コンポーネントの入力ポートに入力データがなくても動作します。ただし、最良の結果を得るには、デバッグ機能を使用する前に入力ポートにデータを入力してください。

1　スクリプトを検証するか、デバッグ・モードに切り替えます。

　行 1 を指す緑色の矢印は、実行の進捗状況を示します。

2　[Task] タブに “Successful compilation” という結果メッセージが表示されることを確認します。

3　次の行に移動するには、ツールバーの [Step] アイコンをクリックします。

　実行中の任意の時点で、変数名と現在の値を検査できます。変数名と現在の値を検査するには、ナビゲータで変数を選択して右クリックします。

❖ ブレークポイントの追加と削除

プロシージャを行ごとにステップ実行する代わりに、選択した行にブレークポイントを追加します。

1　ブレークポイントを追加または削除するには、ブレークポイントを設定する行をクリックします。

2　右クリックし、[Add/Remove breakpoint] を選択します。

❖ ブレークポイントまでのステップ実行

1　ステップごとにツールバーの [Go] アイコンをクリックします。

2　最後のブレークポイントで [Go] アイコンをクリックすると、スクリプトの残りの部分が実行されます。

変数のインライン検査

デバッグ・モードでコードをステップ実行しているとき、またはコードが実行された後、変数の現在の値のインライン検査を実行できます。変数を右クリックすると、変数名と値が表示されます。

ウォッチ・リストでの値のモニタリング

ウォッチ・リストを使用して、コードの実行中における変数値の変化をモニタリングできます。コードをステップ実行するとき、1 つまたは複数の変数で発生する変化をウォッチ・リストで確認できます。

❖ **ウォッチ・リストへの変数の追加**

1 変数を右クリックします。

2 [Add to Watchlist] を選択します。

❖ **ウォッチ・リストからの変数の削除**

1 [Watch List] タブで、変数ローを選択し、右クリックします。

2 [Remove Watch Variable] を選択します。

特殊な JavaScript 機能

実行の中断

JavaScript の実行を中断するには、エディタのツールバーの [Cancel a running script] アイコンをクリックします。

ユーザ定義エラーの作成

エラーを適用してプロジェクトの実行を中断するには、throw("*xx*") 関数を使用します。たとえば、製品の名前 (PR_NAME) が 20 文字より長い場合に実行を停止するには、次のように入力します。

```
if (uLength(IN.PR_NAME) > 20) (
      throw("Product name exceeds maximum length");
)
```

ユーザ定義関数の作成

スクリプト内で関数を定義して、ネスト関数を作成できます。たとえば、次のスクリプトの結果は、変数 *b* の値が 6 になります。

```
var a = 2;
var b = 20;
b = IncA(a);
// end
function IncA (a)
{
     var b = 3;
     a = IncB(b) + a++;
```

```
        return a;
        function IncB(b)
        {
             b = b + 1;
             return b;
        }
    }
```

データ型の変換

Sybase ETL のすべての変数は文字列として表されます。そのため、数値を使用すると予期しない動作が発生することがあります。parseInt() および parseFloat() 関数を使用して、文字列を整数型または浮動型に変換できます。

```
var a = "123";
var b = "22";

a > b

will return FALSE while

parseInt(a) > parseInt(b)
returns TRUE.
```

ファイルのインクルード

外部ファイルをスクリプトにインクルードするには、uScriptLoad(“*filename*”) 関数を使用します。外部ファイルには、関数など、任意の有効な JavaScript 構造体を含めることができるので、再使用可能なコードを作成できます。次に例を示します。

```
uScriptLoad("C:¥scripts¥myfunc.js");
var a = 11;
var b = 2;
var c = 0;
b = gcd(a, b);
// gcd function defined in C:¥scripts¥myfunc.js
```

SQL クエリおよびコマンドの実行

Source、Lookup、Staging、Destination コンポーネントに関連付けられたデータベースに対して SQL クエリまたは一連の SQL コマンドを入力して実行できます。

❖ **SQL クエリの実行**

1 設計ウィンドウでコンポーネントを右クリックし、[Execute SQL Query] を選択します。

2 [Query] フィールドに select 文を入力します。接続されているデータベースの有効な SQL 表記を使用してクエリを構築できます。使用可能なデータベース・スキーマのテーブルとビューを使用してクエリを作成するには、[Database Lookup] アイコンをクリックします。

3 [Execute the query] アイコンをクリックします。

正常に実行されると、[Data Viewer] ウィンドウに結果が表示されます。

❖ **SQL コマンドの実行**

1 設計ウィンドウでコンポーネントを右クリックし、[Execute SQL Commands] を選択します。

2 [Command] フィールドに SQL コマンドを入力します。使用可能なデータベース・スキーマのテーブルとビューを使用してコマンドを作成するには、[Database Lookup] アイコンをクリックします。

3 [Execute the command] アイコンをクリックします。

SQL コマンドが正常に実行されると、メッセージが表示されます。

注意 一連のコマンドを実行しても結果セットは返されません。

パラメータ・セット

プロジェクトを実行するとき、リポジトリに格納されている値ですべてのコンポーネント・プロパティが初期化されます。パラメータ・セットを使用して、これらの値の一部を上書きできます。たとえば、プロジェクトまたはジョブを開発から運用に移行するときに、パラメータ・セットを使用してデータベース接続の設定を変更できます。パラメータ・セットを使用するには、次の手順に従います。

- パラメータとして使用するコンポーネント・プロパティを選択します。
- パラメータ値のセットを格納します。
- 格納されたパラメータ・セットを実行時に割り当てます。

❖ **実行パラメータとしてのコンポーネント・プロパティの選択**

1　プロパティ・ウィンドウで、実行パラメータとして使用するすべてのコンポーネント・プロパティを右クリックし、[External] を選択します。

2　パラメータ・セットを介して式を割り当てるすべてのコンポーネント・プロパティを右クリックし、[Evaluate] を選択します。すべての非表示の値 (タブや CRLF など) を含めます。

3　プロジェクトを保存します。

注意 プロジェクト・パラメータを提供するすべてのコンポーネントに一意の名前を指定してください。プロパティのプロンプトおよび説明も変更できます。プロパティのプロンプトおよび説明を変更する方法の詳細については、「カスタム・プロパティ」(100 ページ) を参照してください。

パラメータ・セットの管理

プロジェクトおよびジョブにパラメータ・セットを割り当てることができます。

[Parameter Set] ウィンドウを開くには、設計ウィンドウまたはナビゲータでプロジェクトまたはジョブを右クリックし、[Parameter Sets] を選択します。

選択したプロジェクトまたはジョブに対して定義されているパラメータ・セットのリストが [Parameter Set] ウィンドウに表示されます。

注意 プロジェクト設計で表示される値がパラメータ・セットに入力する必要のある値と異なるプロパティがあります。このようなプロパティとその値のリストについては、「特殊なプロパティ値」の項を参照してください。

❖ **パラメータ・セットの作成**

1　[Parameter Set] ウィンドウで [Set] - [New] をクリックします。定義済みパラメータのリストがウィンドウに表示され、その現在の値が一番下に表示されます。

2　追加する値で現在の値を上書きします。

3　[Save] をクリックします。

4 パラメータ・セットの名前を入力します。

5 [OK] をクリックします。

❖ **パラメータ・セットの変更**

1 [Parameter Set] ウィンドウでパラメータ・セットを選択します。

2 [Set] - [Open] をクリックします。

3 新しい値で現在のパラメータを上書きします。

4 [Save] をクリックします。

❖ **パラメータ・セットの削除**

1 [Parameter Set] ウィンドウでパラメータ・セットを選択します。

2 [Set] - [Delete] をクリックします。

❖ **パラメータ・セットのコピー**

1 [Parameter Set] ウィンドウでパラメータ・セットを選択します。

2 [Set] - [Copy] をクリックします。

3 新しいパラメータ・セットの名前を入力します。

4 [OK] をクリックします。

❖ **パラメータ・セットを使用したプロジェクトまたはジョブの実行**

1 ナビゲータでプロジェクトまたはジョブを右クリックします。

2 [Execute Project with Parameter Set] または [Job Execute with Parameter Set] を選択します。追加の [Execute] ボタンを含む [Parameter Set] ウィンドウが開きます。

3 リストからパラメータ・セットを選択するか、1 つの実行のパラメータ値を追加します。

4 [Execute] をクリックします。詳細については、「Execution Monitor」(88 ページ) を参照してください。

注意 保存済みのパラメータ・セットが使用できない場合、ウィンドウは自動的に編集モードで開きます。スケジュール・プロジェクトおよびスケジュール・ジョブへのパラメータ・セットの割り当ての詳細については、「ジョブとスケジュール・タスクの管理」(66 ページ) を参照してください。

パラメータ値の割り当て

パラメータの選択

- 1 つのパラメータを選択するには、適切なリスト・ローをクリックします。
- 複数のパラメータを選択するには、目的のロー全体をマウスでドラッグするか、[CTRL] キーを押しながら目的のローを選択します。

パラメータ値の入力または編集

パラメータを選択した後、次の操作を行います。

- 新しい値を入力します。古い値が上書きされます。
- [Value] セルで既存の値を直接編集します。
- [Value] セルのポップアップ・メニューから [Edit] を選択し、値を編集します。[Save] をクリックします。

❖ 複数のプロパティへの同じ値の割り当て

パラメータ・セットはコンポーネント・プロパティに基づいているので、複数のプロパティに同じ値を割り当てる場合があります。

1　値を割り当てるパラメータを選択します。

2　新しい値を入力し、選択したすべての行にその値を使用することを確認します。

あるいは、[Edit Selected values] ボタンをクリックするか、[Edit] またはポップアップ・メニューから [Edit selected] を選択し、値を編集して確認します。

パラメータ・リストのソート

1 つまたは複数のカラムを使用してパラメータ・リストをソートできます。

❖ 1 つのカラムによるパラメータのソート

1　カラム・ヘッダをクリックします。

2　クリックするたびに、昇順、降順、元のソート順に切り替わります。

❖ **複数のカラムによるパラメータのソート**

1 1 番目のカラム・ヘッダを何度かクリックして、適切な順序にソートします。

2 [Ctrl] キーを押しながら 2 番目のカラム・ヘッダをクリックしてカラムをソートします。

特殊なプロパティ値

プロジェクト設計で表示される値がパラメータ・セットに入力する必要のある値と異なるプロパティがあります。

チェックボックス

プロジェクト設計でチェックボックスによって表されるプロパティには、値として 0 (非アクティブ) または 1 (アクティブ) を指定する必要があります。

式

パラメータ・セットで変数値を使用するには、設計ウィンドウの場合と同様に式を入力します。[Eval] カラムは、プロパティが式で有効かどうかを示します。詳細については、「角カッコ表記」(72 ページ) を参照してください。

式は、非表示の文字 (タブや CRLF など) を含む値を設定する場合に特に必要です。プロジェクトを設計するときは、これらのプロパティに [Evaluate] オプションを設定する必要があります。

注意 式は [Parameter Set] ウィンドウでは検証できません。

ドロップダウン・メニュー

一部のメニューでは、基になるパラメータ値が表示されません。これらの値をパラメータ・セットで割り当てるには、[Evaluate] オプションの設定が必要になる場合があります。

次の表は、どの値 (値) が表示値 (プロンプト) に対応しているかと、[Evaluate] オプションを有効にする必要があるかどうかを示しています。

コンポーネント	プロパティ	プロンプト	値	評価
DB コンポーネント	インタフェース	ODBC	dbodbc	
		Sybase	dbsybase	
		Oracle	dboracle	
		IBM DB/2	dbdb2	
		SQLite Persistent	dbpersistent	
		OLE DB	dbole	
テキスト・コンポーネント	ロー・デリミタ	位置		
		LF	[uChr(10)]	x
		CR	[uChr(13)]	x
		CRLF	[uConcat(uChr(13),uChr(10))]	x
	カラム・デリミタ	位置		
		タブ	[uChr(9)]	x
		カンマ	,	
		セミコロン	;	
	カラム引用符	なし		
		一重引用符	'	
		二重引用符	"	

複数のエンジンの使用によるジョブ実行時間の短縮

グリッド・アーキテクチャでは、複数の分散エンジンでのプロジェクトの並列実行を使用することにより、ジョブの実行時間が短縮されます。

このスケーラビリティを活用するには、次の作業を行う必要があります。

- 複数のグリッド・エンジンのインストール。
- グリッド・エンジンの登録。
- マルチエンジン実行用のジョブの準備。

注意 このマニュアルでは、グリッド・エンジンと ETL サーバという用語は同じ意味で使用されています。

❖ **グリッド・エンジンの登録**

グリッド・エンジンをインストールした後、特殊なリポジトリに登録できます。そのリポジトリからマルチエンジン・ジョブを実行すると、それらのエンジンですべてのプロジェクトが実行されます。

グリッド・エンジンを登録するには、[Tools] - [Engine Manager] を選択します。複数のリポジトリへの接続が開いている場合、いずれかの接続を選択します。選択したリポジトリに既に登録されているエンジンのリストが [Engine Manager] ウィンドウに表示されます。

登録されているエンジンのプロパティを次に示します。

- Name：エンジンのユーザ定義の名前。
- Host：エンジン・ホストの名前または IP アドレス。
- Port：エンジンが受信しているポートの番号。
- Base Rank：エンジンのユーザ定義のランク付け。ジョブでは、最初に最高ランクのエンジンでのプロジェクト実行が試行されます。
- Description：サーバの説明。

個々または複数の ETL サーバを手動で登録できます。

❖ **グリッド・エンジンの手動登録**

1 [Engine] - [New] を選択するか、[New Insert] アイコンをクリックします。

2 目的の値を入力します。

3 [OK] をクリックします。

❖ **複数のエンジンの登録**

1 [Engine] - [New network search] を選択します。[New Engines] ウィンドウに、登録情報と追加情報に加えて、エンジンのオンライン・ステータスが表示されます。

2 登録するエンジンを選択します。

3 [Add] をクリックします。

注意 グリッド・エンジンのリストを更新して再ロードするには、ツールバーの [Refresh] アイコンをクリックするか、[F5] キーを押します。

❖ **エンジン登録の変更**

1　リストでエンジンを選択し、[Engine] - [Edit] を選択するか、[Edit selected engine] アイコンをクリックします。または、リスト内のエンジンをダブルクリックします。

2　新しい値で現在の値を上書きします。

3　[OK] をクリックします。

❖ **エンジン登録の削除**

1　削除するエンジンを選択します。

2　[Engine] - [Delete] を選択するか、ツールバーの [Delete selected engine] アイコンをクリックします。

マルチエンジン・ジョブの定義

並列グリッド・アーキテクチャを使用して、複数のエンジンでジョブを実行できます。一般的なマルチエンジン・ジョブには、相互の依存性がない (または少ない) 複数のプロジェクトが含まれます。したがって、プロジェクトを複数のエンジンで同時に実行できます。

1　ジョブをダブルクリックします。

2　設計ウィンドウで右クリックし、[MultiEngine Execution] を選択します。

実行中、ジョブは登録済みのエンジンを使用してプロジェクトを分散します。

マルチエンジン・ジョブの実行

マルチエンジン・ジョブを実行するには、ナビゲータでジョブを右クリックし、[Job Execute] を選択します。または、Runtime Manager でジョブの実行をスケジュールします。

Engine Monitor

Engine Monitor には、環境内で使用可能なすべてのグリッド・エンジンに関する情報 (Engine Manager で実行または登録されているかどうか) が表示されます。

注意 マルチエンジン・ジョブの実行に使用できるのは、Engine Manager で登録されているエンジンだけです。

使用可能なグリッド・エンジンに関する情報を表示するには、[Tools] - [Engine Monitor] を選択します。すべてのエンジンに関する詳細情報が Engine Monitor に表示されます。更新間隔 (秒) を [Update Interval] フィールドで指定できます。デフォルト値は 5 秒です。

Execution Monitor

Execution Monitor には、現在のジョブまたはプロジェクトのプロパティが表示されます。Execution Monitor を表示するには、次の手順に従います。

1 実行するプロジェクトを選択します。

2 ツールバーの [Execute] をクリックします。

[Execution Monitor] ウィンドウは、[Job] と [Projects] の 2 つのパネルに分かれています。

[Execution Monitor] ウィンドウの上部には、現在実行中のジョブのプロパティが表示されます。

プロパティ	説明
Name	ジョブまたはプロジェクトの名前
State	現在のジョブ実行ステータス
Start	開始日時
Stop	中止日時
Message	エラー・メッセージ

[Projects] リストには、ジョブ内の各プロジェクトの行が含まれています。

プロパティ	説明
Name	プロジェクト名
State	現在のプロジェクト実行ステータス
Start	開始日時
Stop	中止日時
Engine Name	実行エンジンの名前
Engine Host	実行エンジンのホスト
Engine Port	実行エンジンのポート
Message	エラー・メッセージ

実行結果の保存またはコピー

HTML ファイルに実行したプロジェクトの結果を保存するには、[Save Results] ボタンをクリックします。

メモ帳や Microsoft Word など他のアプリケーションに実行したジョブまたはプロジェクトの結果をコピーして貼り付けるには、Execution Monitor に表示されているジョブまたはプロジェクトを右クリックして [コピー] を選択します。

ジョブ実行のキャンセル

ジョブ実行をキャンセルするには、[Execution Monitor] ウィンドウで [Cancel Execution] をクリックします。

グリッド・エンジンによって、実行中のプロジェクトのキャンセルが試行されます。実行を待機しているプロジェクトは開始されません。

パフォーマンス・データの分析

Sybase ETL では、ジョブとプロジェクトの実行中にパフォーマンス関連データが収集され、リポジトリ・テーブルに保存されます。

❖ **パフォーマンス・データの収集**

1　パフォーマンス関連データを収集してリポジトリに保存するには、[File] - [Preferences] - [Performance Logging] を選択します。

2　ログ・レベル・リストで [1] を選択します。

❖ **パフォーマンス・データの表示**

1 **選択したプロジェクトまたはジョブに関するパフォーマンス・データの概要を表示する**

ナビゲータでプロジェクトまたはジョブを右クリックして、[Performance Data] を選択します。または、プロジェクトを開き、設計ウィンドウの任意の場所を右クリックして、[Performance Data] を選択します。

[Performance Data] ウィンドウが表示され、選択したプロジェクトまたはジョブに関する実行内容の詳細が [Overview] タブに表示されます。デフォルトでは、最新の月に行われた実行に関するパフォーマンス・データが表示されます。表示される実行日の範囲を変更するには、ツールバーの [Execution Date Range] の値を変更します。

注意 一度に 1 つのプロジェクトまたはジョブに関するパフォーマンス・データのみを表示できます。

任意の実行に関するパフォーマンス・データを削除するには、特定のローを選択して、ツールバーの [Delete selected performance log entries] アイコンをクリックします。または、[Ctrl+D] キーを押します。

警告！ この操作を行うと、選択した実行に関するパフォーマンス・データがパフォーマンス・ログから完全に削除されます。 削除したデータは復元できません。

2 **プロジェクトのパフォーマンス・データを表示する**

プロジェクトに関するパフォーマンスの詳細を表示するには、[Overview] タブでローを選択して、ツールバーの [Drill down in performance data] アイコンをクリックします。または、表示されたチャートで、選択したローまたは対応するバーをダブルクリックします。新しいタブが表示され、そこに、コンポーネントの名前、継続期間、読み込まれたレコード、書き込まれたレコードなどの詳細が表示されます。

注意 IQ Loader DB via Insert Location コンポーネントと IQ Loader File via Load Table コンポーネントを使用して、プロジェクトのパフォーマンス・データを表示する場合、読み込まれたレコード数と書き込まれたレコード数がパフォーマンス・テーブルに正しく表示されないことがあります。この情報は ETL には使用できません。IQ では、ファイルまたはリモート・データベースからデータが直接ロードされるからです。

選択したコンポーネントに関するパフォーマンスの詳細を表示するには、コンポーネントの詳細を表示しているタブでローを選択して、ツールバーの [Drill down in performance data] アイコンをクリックします。または、表示されたチャートで、選択したローまたは対応するバーをダブルクリックします。イベント名、継続期間、読み込まれたレコード、書き込まれたレコードなどのコンポーネントに関するパフォーマンスの詳細が表示されます。

注意 Load Project コンポーネントの詳細は表示できません。このコンポーネントは、実行前にエンジンがプロジェクトをロードする時間を表しています。

上位レベルのパフォーマンスの詳細に戻るには、ツールバーの [Drill up in performance data] アイコンをクリックします。[Overview] タブに戻るには、ツールバーの [Return to performance data overview] アイコンをクリックするか、[Navigation] - [Return to overview] を選択します。

3　**ジョブに関するパフォーマンス・データを表示する**

ジョブに関するパフォーマンスの詳細を表示するには、[Overview] タブでローを選択して、ツールバーの [Drill down in performance data] アイコンをクリックします。または、表示されたチャートで、選択したローまたは対応するバーをダブルクリックします。新しいタブが表示され、そこに、プロジェクトの名前、継続期間、読み込まれたレコード、書き込まれたレコードなどの詳細が表示されます。

選択したプロジェクトに関するパフォーマンスの詳細を表示するには、ローを選択して、ツールバーの [Drill down in performance data] アイコンをクリックします。または、表示されたチャートで、選択したローまたは対応するバーをダブルクリックします。コンポーネント名、継続期間、読み込まれたレコード、書き込まれた

レコードなどのプロジェクトに関するパフォーマンスの詳細が表示されます。

注意 Load Project コンポーネントの詳細は表示できません。このコンポーネントは、実行前にエンジンがプロジェクトをロードする時間を表しています。

上位レベルのパフォーマンスの詳細に戻るには、ツールバーの [Drill up in performance data] アイコンをクリックします。[Overview] タブに戻るには、ツールバーの [Return to performance data overview] アイコンをクリックするか、[Navigation] - [Return to overview] を選択します。

注意 選択したコンポーネントをプロジェクトで検索するには、[Tools] - [Show component] を選択します。または、ツールバーの [Show selected component] アイコンをクリックします。

パフォーマンス・データのクエリをリセットする – パフォーマンス・データのクエリは、*HKEY_CURRENT_USER¥Software¥JavaSoft¥Prefs¥sybase¥sybetl¥* の下のシステム・レジストリに保存されます。[Performance Data] を使用中にエラーが発生した場合、これらのクエリをデフォルト値にリセットする必要があります。変更したクエリを任意の時点でデフォルト値にリセットするには、レジストリの [Performance Data Query] キーを削除して、アプリケーションを再起動します。

警告！ やむを得ない場合を除いて、クエリは変更しないでください。

❖ **パフォーマンス・データの検索**

1 パフォーマンス・データ・テーブルでローを選択します。

2 [Ctrl+F] キーを押して、検索ウィンドウを開きます。検索条件を [Find] フィールドに入力します。

テーブルで、検索されたデータが強調表示されます。

❖ パフォーマンス・データの出力

1　[Tools] - [Generate Report] を選択します。

2　出力する詳細レベルを選択して、送信先ファイルを選択します。使用できるオプションは次のとおりです。

- 概要
- ジョブのパフォーマンス (ジョブに関するパフォーマンス・データ・レポートを出力する場合にのみ表示)
- プロジェクトのパフォーマンス
- コンポーネントのパフォーマンス

3　生成するレポートを格納する送信先ファイルのパスを入力するか、デフォルト値を受け入れます。

4　レポート・データを [Overview] タブで選択した実行に限定するには、[Only selected executions] チェックボックスを選択します。

注意　[Overview] タブで実行を選択しない場合、[Only selected executions] チェックボックスは表示されません。

5　[Generate] をクリックします。レポートの生成に成功した場合は、メッセージが表示されます。[Yes] をクリックして、パフォーマンス・データ・レポートを表示します。

レポートには、テーブルとグラフィカル・データの両方が表示されます。また、レポートの対応する項にリンクしている目次があります。各レポートには、出力するように指定した詳細レベルに応じて次の項があります。

- 概要
- ジョブのパフォーマンス (ジョブに関するパフォーマンス・データ・レポートにのみ表示)
- プロジェクトのパフォーマンス
- コンポーネントのパフォーマンス

パフォーマンスの概要の項では、各プロジェクトまたはジョブの実行の継続期間が表示されます。プロジェクト、コンポーネント、ジョブのパフォーマンスの各項では、選択した実行に関するパフォーマンス・データのテーブルとチャートが表示され、細分化されたデータの複数のレベルがそれぞれに示されます。

パフォーマンス・データ・モデルと内容

パフォーマンス・データは、TRON_PERFORMANCE という名前の、単一で非正規化されたリポジトリ・テーブルに保存されます。このテーブルは、パフォーマンス・データを収集する場合の問い合わせ先になります。この項では表示される情報について説明します。

イベント

パフォーマンス・ログはイベントに基づいています。各イベントには、開始時刻 (完全タイムスタンプ、日付、時刻の 3 形式) と継続期間 (単位：ミリ秒) が保存されます。イベントの説明は、クラス、名前、テキストで構成されています。一部のイベントには結果があります (成功または失敗など)。また、イベントを報告したエンジンに関する情報があります。

次のリストで、報告されたイベントについて簡単に説明します。

クラス	名前	説明
control	execute job	ジョブの総実行時間 (ジョブ実行ごとの 1 レコード、属性 PRF_JOB_DURATION での継続期間 (単位：ミリ秒))。
init	load job	リポジトリからジョブ定義を取得する。
control	execute project	プロジェクトの総実行時間 (プロジェクト実行ごとの 1 レコード、属性 PRF_PRJ_DURATION での継続期間 (単位：ミリ秒))。
init	load project	リポジトリからプロジェクト定義を取得する。
init	create	プロジェクトとコンポーネントのインスタンスを作成する。
init	configure	プロジェクトとコンポーネントのインスタンスを設定する。
perform	prepare	コンポーネントの前処理を実行する。
perform	process	コンポーネントのステップを実行する。
perform	finish	コンポーネントの後処理を実行する。
perform	read	コンポーネントの入力ポートへのデータを取得する。
perform	write	コンポーネントの出力ポートからのデータをプッシュする。

注意 プロジェクトの総実行時間は、分配され、マルチスレッドで実行されるために、関与しているすべてのコンポーネントの実行時間の合計より大幅に短くなる場合があります。

一般情報

プロジェクトまたはジョブの個々の実行は、グローバルなユニーク ID によって識別されます。実行の開始時刻は、完全タイムスタンプ、日付、時刻の 3 形式で表示されます。追加情報は、実行を開始するアカウントと、プロジェクトまたはジョブが格納されているリポジトリに関するものです。

注意 実行とイベントの開始時刻は、グリニッジ標準時 (GMT) とも呼ばれる万国標準時 (UTC) でログインされます。

ジョブの実行情報

ジョブには、ID、バージョン (変更した日付)、名前があります。単一のジョブ実行イベントには、ジョブの継続期間がミリ秒単位で保存されます。ジョブのコンポーネント (プロジェクト) は、ID クラス、バージョンで表されます。

プロジェクトの実行情報 (ジョブ・プロジェクトを含む)

実行された各プロジェクトには、ID、バージョン (変更された日付) 名前、グローバルなユニーク実行 ID が付与されます。単一のプロジェクト実行イベントには、プロジェクトの継続期間がミリ秒単位で保存されます。

プロジェクトのコンポーネントは、ID、名前、クラス、型、バージョンで表されます。プロセス・イベントでは、ステップ数と処理されたレコード容量が提供されます。

ポート・イベントでは、ID、名前、クラス、型、入力ブロックまたは出力ブロック、入力レコードまたは出力レコードが提供されます。

第 5 章

コンポーネント

この章では、さまざまな Sybase ETL コンポーネントの詳細について説明します。

概要

Sybase ETL コンポーネントは、プロジェクトおよびジョブの作成に使用します。これらのコンポーネントは、コンポーネント・ストアにあります。プロジェクト・コンポーネントには次のものが含まれます。

- Source コンポーネント – 変換ストリームのデータを提供します。プロジェクトは 1 つまたは複数の Source コンポーネントとともに起動する必要があります。Source コンポーネントには IN ポートがなく、1 つまたは複数の OUT ポートがあります。
- Transformation コンポーネント、Lookup コンポーネント、および Staging コンポーネント – 特定の変換を変換ストリームのデータに適用します。これらのコンポーネントの型には、IN ポートと OUT ポートが少なくとも 1 つずつあります。
- Destination コンポーネント (データ・シンクともいいます) – データを特定のターゲットに書き込みます。Destination コンポーネントには 1 つの IN ポートがありますが、OUT ポートはありません。

- Loader コンポーネント – 変換を実行しないで、ソース・データベースまたはファイルからデータを抽出して IQ データベースにロードします。

注意 機能的な観点では異なりますが、すべてのコンポーネントが共通の概念を共有します。

コンポーネント・プロパティの設定

各コンポーネントは固有のタスク専用で、タスク固有の機能を備えています。ただし、どのコンポーネントも同じ手順に従ってプロパティを設定します。

プロジェクト内でコンポーネントを使用する前に、必要なプロパティを設定する必要があります。プロパティは次のウィンドウで設定できます。

- コンポーネントを設計ウィンドウに追加すると表示される設定ウィンドウ。
- プロジェクトに追加したコンポーネントのプロパティ・ウィンドウ。

SBN 式の評価

評価プロパティを使用して、プロパティ値を使用する前に評価される角カッコ表記 (SBN) 式を使用することができます。詳細については、「角カッコ表記」(72 ページ) を参照してください。

注意 一部のプロパティについては、評価プロパティがデフォルトで選択されています。

❖ **評価プロパティの変更**

次のいずれかの方法で、評価プロパティを変更することができます。

1　プロパティ・ウィンドウで、プロパティの [Eval] オプションを選択します。

2　右クリックして [Evaluate] を選択します。

注意 [Password] プロパティの [Eval] オプションを選択すると、プレーン・テキストとして値が表示されます。

プロパティの暗号化

プロパティ値はリポジトリに格納されています。ただし、Sybase ETL リポジトリのエントリは暗号化されておらず、人間が読める文字セットで表現されています。

暗号化プロパティを切り替えるには、[Encrypt] チェックボックスをオンにするか、プロパティ・ウィンドウでプロパティを右クリックして、[Encrypt] を選択します。

プロパティ参照変数

単純な値を持つプロパティには、コンポーネントの式およびプロシージャで参照できる変数が関連付けられています。[Evaluate] が設定されている場合、変数には必ず、評価されるプロパティの現在値が含まれています。

- 変数名を表示するには、プロパティ・ウィンドウでプロパティを右クリックして、[Reference Variable] を選択します。
- JavaScript Debugger で変換規則を設定するときは、[Variable] - [Parameter] をクリックして、[Reference Variables] にアクセスします。

カスタム・プロパティ

コンポーネントにカスタム・プロパティを追加できます。その他のプロパティと同様、これらのプロパティはコンポーネントの式またはプロシージャで参照できる変数に関連付けられています。実行時に評価されるか、パラメータ・セットを通じて割り当てられる式を含めることができます。

❖ **カスタム・プロパティの追加**

1 設計ウィンドウで、コンポーネントを選択します。

2 プロパティ・ウィンドウで右クリックし、[Add] を選択します。

3 プロパティの名前を入力します。予約されている JavaScript キーワードは使用しないでください。詳細については、「変数」(70 ページ) を参照してください。コンポーネント内では、このプロパティは変数 REF.< **プロパティの名前** > を使用して参照されます。

4 プロンプト・フィールドに値を入力します。これは、プロパティ・ウィンドウに表示されるラベルです。

5 説明フィールドに値を入力します。この説明はプロパティのヒントに表示されます。

6 [OK] をクリックします。

❖ **カスタム・プロパティの削除**

1 プロパティ・ウィンドウで、削除するカスタム・プロパティを選択します。

2 右クリックし、[Remove] を選択します。[OK] をクリックして確定します。

注意 必ず、関連付けられている変数へのすべての参照を削除してください。

コンポーネントへの説明の入力

コンポーネントに名前と説明を割り当てることができます。コンポーネント上にマウスを移動すると、ヒントに説明と名前が表示されます。名前はコンポーネントの上部にも表示されます。

❖ **コンポーネントへの説明の追加**

1　設計ウィンドウでコンポーネントを右クリックし、[Description] を選択します。

2　コンポーネントの名前と説明を入力します。HTML フォーマット・タグを使用して説明をフォーマットします。

ポート構造の設定

コンポーネントには、データを受け取る IN ポートと、処理後にデータを渡す OUT ポートがあります。あるコンポーネントのステップを実行すると、OUT ポートを通じて転送されたデータは次のコンポーネントの IN ポートに渡されます。

ポート構造の管理

非構造化ポートと構造化ポート間に接続を追加すると、非構造化ポートは構造化ポートの構造を継承します。

ポート構造に追加属性を追加できます。

プロジェクトにコンポーネントを追加すると、コンポーネントの各ポートの色はそのコンポーネントのステータスを示します。

- 緑 – コンポーネントが正しく設定されています。
- 黄 – ポート構造は定義されていますが、1 つまたは複数の必須プロパティが定義されていません。
- 赤 – ポート構造が未定義で、1 つまたは複数の必須プロパティも定義されていません。

注意　色の識別が困難なユーザを支援するために、[File] - [Preference] ウィンドウの [Use enhanced color accessibility] オプションを選択してコンポーネントの各ポートの色を変更することができます。詳細については、「設定のカスタマイズ」(22 ページ) を参照してください。

ポート構造の変更

❖ ポート構造の変更

1 設計ウィンドウでポートを右クリックし、[Edit Structure] を選択します。

2 [Structure Viewer] ウィンドウで必要な変更を行い、[Save] をクリックします。

注意 変更できないポート構造については、[Edit Structure] オプションを使用することはできません。[View Structure] オプションを使用してポート構造を表示することのみ可能です。

❖ ポート属性の追加

1 設計ウィンドウでポートを右クリックし、[Edit Structure] を選択します。

2 [Structure Viewer] ウィンドウで [Actions] - [Add] を選択するか、属性を右クリックして [Add] を選択します。

3 新しい属性の名前を入力します。ポート属性の名前は英文字で始める必要があり、英数字のみを使用できます。予約されている JavaScript キーワードは使用しないでください。詳細については、「変数」(70 ページ) を参照してください。

[Populate Attribute] オプションを選択して、複数のポート構造に属性を追加します。選択した接続に参加しているすべてのポート構造に新しい属性が追加され、自動的にマッピングされます。[OK] をクリックします。

❖ ポート属性の削除

1 設計ウィンドウでポートを右クリックし、[Edit Structure] を選択します。

2 [Structure Viewer] ウィンドウで [Actions] - [Remove] を選択するか、属性を右クリックして [Remove] を選択します。

❖ ポート属性の変更

1 設計ウィンドウでポートを右クリックし、[Edit Structure] を選択します。

2 [Structure Viewer] ウィンドウで属性設定を変更し、[Save] をクリックします。

❖ **その他のポートからのポート構造のコピー**

現在のプロジェクトのその他の使用可能なポートに基づいて、ポートにポート構造を割り当てることができます。

1　設計ウィンドウで、新しい構造を割り当てるポートを選択します。

2　右クリックし、[Assign Structure] - [Copy Structure] を選択します。

同じコンポーネントの他のポートからポート構造をコピーできます。[Copy Structure] を選択して、現在のプロジェクトのその他のポート構造をコピーすることもできます。

[Copy Structure] を選択すると、ウィンドウに現在のプロジェクトの概要グラフが表示されます。プロジェクト内の使用可能なポートのいずれかを選択できます。

❖ **新しいポート構造のソースとして使用するポートの選択**

1　ソースとして使用するポートをクリックします。選択したポートの属性構造は、ウィンドウの下部に表示されます。

2　[Apply] をクリックします。

コンポーネントのシミュレーション

コンポーネントの初期化

1　設計ウィンドウで、コンポーネントを選択します。

2　右クリックし、[Initialize] か [Initialize and Step] を選択します。

注意　シミュレーション時にコンポーネントの既存のプロパティ設定のいずれかを変更した場合は、コンポーネントを再初期化してから次に進んでください。

コンポーネントのステップを複数回実行

異なるプロパティ設定でのコンポーネントの動作をプレビューするために、シミュレーション時にコンポーネントのステップを複数回実行できます。ステップを繰り返しても、ダウンストリーム・コンポーネントに渡されるレコードの数は増えません。

❖ コンポーネントのステップを複数回実行

1 設計ウィンドウで、コンポーネントをクリックします。

2 プロパティ・ウィンドウで、変換規則またはプロパティ設定を変更します。

3 コンポーネントを右クリックし、[Initialize] を選択します。

4 コンポーネントを右クリックし、[Step] を選択します。

5 手順 2 に戻ります。

コンポーネントのステップを繰り返し実行しても、IN ポートでは各ステップで同じレコード・セットが再処理されて出力ポートに転送されるだけで、OUT ポートでのレコード・セットの数は増えません。多くのコンポーネントの場合、コンポーネント・ウィンドウ内から、または Sybase ETL Development ウィンドウで右クリックすると表示されるメニューを使用して、ステップ実行を行うことができます。

変換結果のプレビュー

シミュレーション・モードで操作する場合、データ・ソースの結果セット全体を Source コンポーネントまたは Staging コンポーネントのクエリによって定義することで複数のデータ・ブロックに分けることができます。データ・ブロックにはレコードのサブセットが含まれます。各サブセットのレコードの数は、コンポーネントのプロパティ・ウィンドウに表示される [Read Block Size] パラメータに関連しています。プロジェクトをステップ実行する場合のパフォーマンスを向上させるには、小さな数値の [Read Block Size] パラメータを選択します。

❖ 変換結果のプレビュー

1 プレビューするコンポーネントを含んでいるプロジェクトをステップ実行します。

2 コンポーネント、ポート、または接続リンクを右クリックして [Preview] を選択します。

コンポーネントに複数のポートがある場合は、最初に、変換結果をプレビューするためのポートを選択します。

データベース接続の設定

次のウィンドウでデータベース接続パラメータを指定することができます。

- コンポーネントを設計ウィンドウに追加するか、既存のプロジェクト・コンポーネントをダブルクリックすると表示される [Database Configuration] ウィンドウ。
- 設計ウィンドウのプロジェクト・コンポーネントを選択すると表示されるプロパティ・ウィンドウ。

データベース接続パラメータは次のとおりです。

- [Interface] – 使用するメソッドまたはドライバを選択して、送信元または送信先データベースに接続します。特別なデータベース・オプションを設定するには、[Database options] アイコンをクリックします。

  コンポーネントに使用可能なインタフェースは、次のとおりです。

  - Sybase
  - ODBC

    注意 ODBC ドライバを Sybase ETL Development と同じコンピュータにインストールし、システム・データ・ソース名 (DSN) をターゲットに定義する必要があります。ETL サーバ上でプロジェクトを実行する場合は、ETL サーバが適切な ODBC ドライバおよび DSN にアクセスできる必要もあります。

  - OLE DB
  - Oracle
  - DB2

- SQLite Persistent

  注意 SQLite Persistent は、テスト環境専用で使用します。運用環境では使用しないでください。

注意 Sybase Open Client™ を Sybase ETL Development と同じコンピュータにインストールし、Windows の場合は *%SYBASE%¥ini¥sql.ini* ファイルに、UNIX と Linux の場合は *$SYBASE/interfaces* ファイルに、データベース・サーバを定義する必要があります。ETL サーバ上でプロジェクトを実行する場合は、ETL サーバが Open Client ライブラリにアクセスできる必要もあります。

- [Host Name] – 送信元または送信先データベースを選択します。ホスト名リストに表示されるオプションは、選択したインタフェースによって異なります。

注意 SQLite Persistent インタフェースを選択した場合は、既存または新しいデータベース・ファイル名を入力できます。詳細については、「SQLite データベースへの接続」(314 ページ) を参照してください。

- [Database user name and password] – 認証されたデータベース・ユーザ名およびパスワードを指定します。
- [Database name] – 送信元または送信先データベースとして使用するデータベースを指定します。
- [Database schema] – スキーマまたは所有者を指定して、そのスキーマで表示されるオブジェクトを制限し、新しいテーブルを作成します。
- [Standardize Data Formats] – 異なるフォーマットをサポートしているシステム間で Sybase ETL が移動できるように、受信する日付および番号情報を標準フォーマットに変換します。
- [Database options] – このオプションを選択して、パフォーマンスのデフォルトを上書きし、一部のトランザクションの動作を制御します。データベース・オプションのリストについては、「インタフェース固有のデータベース・オプション」(305 ページ) を参照してください。

Source コンポーネント

Source コンポーネントは変換ストリームのデータを提供します。プロジェクトは 1 つまたは複数の Source コンポーネントとともに起動する必要があります。Source コンポーネントには IN ポートがなく、1 つまたは複数の OUT ポートがあります。

コンポーネント	説明
DB Data Provider Full Load	ODBC 接続またはネイティブ・ドライバ (DB2、Oracle、Sybase) によってアクセス可能なデータ・ソースからデータを抽出します。
DB Data Provider Index Load	インクリメンタル・データ・ロードを実行します。インクリメンタル・ロードは、昇順の値が含まれる [Ascending Index] 属性によって制御されます。
Text Data Provider	テキスト・ファイルからテーブルに構造化データを読み取って変換します。
XML via SQL Data Provider	階層 XML データをリレーショナル・スキーマにロードします。

DB Data Provider Full Load

DB Data Provider Full Load は ODBC 接続またはネイティブ・ドライバ (DB2、Oracle、Sybase) によってアクセス可能なデータ・ソースからデータを抽出する Source コンポーネントです。1 つの出力ポートの構造は、クエリの結果セットの構造を反映します。

DB DataProvider Full Load コンポーネントの設定

1　設計ウィンドウに DB DataProvider Full Load コンポーネントをドラッグします。[Database Configuration] ウィンドウが表示されます。あるいは、その設定ウィンドウを開くために、コンポーネントを選択し、プロパティ・ウィンドウのプロパティ・アイコンをクリックします。

2　接続パラメータを入力します。詳細については、「DataProvider Full Load プロパティ・リスト」(108 ページ) を参照してください。

注意　有効なインタフェースとホスト名を追加する必要があります。

3 [Query] アイコンをクリックします。データ・ソースからデータ・セットを取得するには、クエリを作成して保存します。

[Query] ウィンドウで簡単なクエリを直接作成するか、[Query Designer] アイコンをクリックして Query Designer を開き、クエリを生成します。詳細については、「Query Designer」(59 ページ) を参照してください。

4 [Finish] をクリックします。

5 プロパティ・ウィンドウで、その他のオプション・プロパティおよびデータベース・オプションを指定します。

❖ **ポート構造の更新**

データベースが変更されたポート構造を更新するには、次の手順に従います。

1 DB DataProvider Full Load コンポーネントを右クリックします。

2 [Reconfigure] を選択します。

[Reconfigure] オプションを選択すると、データベース・スキーマが変更されている場合、コンポーネントの設定が更新されます。現在の接続が閉じ、データベースへの新しい接続が開かれ、クエリのメタデータが読み取られ、ポート構造の更新が適用されます。

DataProvider Full Load プロパティ・リスト

DataProvider Full Load プロパティ・リストには、プロパティ・ウィンドウで定義が必要な接続パラメータおよび他の項目を示します。必須のプロパティは、**太字**テキストで表示されます。

必須プロパティ

プロパティ	説明
[Interface]	データ・ソースへの接続に使用するメソッドまたはドライバを識別します。
[Host Name]	データ・ソースを識別します。ホスト名リストにあるオプションは、選択したインタフェースによって異なります。
[Query]	[Query] アイコンをクリックして、データ・ソースから情報を取得するクエリを作成できるウィンドウを開きます。 [Query] ウィンドウを使用して簡単なクエリを作成するか、[Query Designer] アイコンをクリックして Query Designer を開き、クエリを生成します。詳細については、「Query Designer」(59 ページ) を参照してください。

オプション・プロパティ

プロパティ	説明
[User and Password]	認証されたデータベース・ユーザを識別する組み合わせに使用され、認証されていないアクセスからデータベースを保護します。
[Read Block Size]	1 つのステップでコンポーネントが取得するレコードの数を決定します。
[Pre-processing SQL]	[Pre-Processing SQL] アイコンをクリックすると、コンポーネントの初期化中に実行されるクエリを作成できるウィンドウが開きます。 クエリには 1 つまたは複数の SQL 文を含めることができます。複数の文を使用する場合は、セミコロン (;) で区切ります。
[Post-Processing SQL]	[Post-Processing SQL] アイコンをクリックして、すべてのコンポーネントの実行後に実行されるクエリを作成できるウィンドウを開きます。 クエリには 1 つまたは複数の SQL 文を含めることができます。複数の文を使用する場合は、セミコロン (;) で区切ります。
[Database]	データ・ソースとして使用するデータベースを識別します。 このオプションを選択すると、適切なインタフェース、場合によっては適切なユーザ ID およびパスワードを選択する必要があります。

プロパティ	説明
[Schema]	データ・ソースとして使用するスキーマ／所有者を識別します。表示されたオブジェクトは適切に制限され、新しいテーブルがそのスキーマに作成されます。
[Standardize Data Format]	異なるフォーマットをサポートしているシステム間で Sybase ETL が移動できるように、受信する日付および番号情報を標準フォーマットに変換します。 日付は、年、月、日、時、分、秒、秒の小数点以下が含まれるフォーマット `YYYY-MM-DD hh:mm:ss.s` に変換されます。 例を示します。 `2005-12-01 16:40:59.123` 数値は、小数点として「.」を使用して変換されます。
[Database Options]	[Database Options] アイコンをクリックすると、パフォーマンスのデフォルトを上書きし、一部のトランザクションの動作を制御するオプションを設定できるウィンドウが開きます。 詳細については、「データベース接続の設定」(105 ページ) を参照してください。

Data Provider Full Load デモ

Sybase ETL には、Data Provider Full Load コンポーネントのデモが複数含まれています。これらのデモは、[Help] メニューの Flash デモ、および DemoRepository のサンプル・プロジェクトとして使用できます。

Flash デモ

Flash デモを実行するには、[Help] - [Demonstrations] - [Source] をクリックして、[DB Data Provider - Full Load] を選択します。

DemoRepository

ナビゲータで、[Repository] - [TRANSFORMER.transformer.Repository] - [Projects] をクリックして、次を選択します。

- [Demo Transfer German Customers]
- [Demo Transfer German Products]
- [Demo Transfer German Sales]
- [Demo Transfer U.S. Customers]
- [Demo Transfer U.S. Products]

DB Data Provider Index Load

DB Data Provider Index Load は、昇順インデックス値に基づいてインクリメンタル・データ・ロードを実行する Source コンポーネントです。実行時、DB Data Provider Index Load は以前に抽出したデータ・レコードを無視します。

DB Data Provider Index Load を使用して、定期的にソース変更を追跡するインクリメンタル・ロードを実行します。

シミュレーション・シーケンスへの影響

- [Read Block Size] 値は 1 つのシミュレーション・ステップでロードされるレコードの数に影響を与えます。
- シミュレーション時にプロジェクトが実行されても、リポジトリ内の Load Index の値は更新されません。

DB DataProvider Index Load コンポーネントの設定

1 設計ウィンドウに DB DataProvider Index Load コンポーネントをドラッグします。[Database Configuration] ウィンドウが表示されます。あるいは、その設定ウィンドウを開くために、コンポーネントを選択し、プロパティ・ウィンドウのプロパティ・アイコンをクリックします。

2 接続パラメータを追加します。特定のフィールドの要件については、「DataProvider Index Load プロパティ・リスト」(113 ページ) を参照してください。

注意 有効なインタフェースとホスト名を追加する必要があります。

3 [Finish] をクリックします。

4 プロパティ・ウィンドウで [Ascending Index] アイコンをクリックして、データベース・オブジェクトのリストから昇順インデックス属性を選択します。

5 [Query] アイコンをクリックします。データ・ソースからデータ・セットを取得するには、クエリを作成して保存します。

[Query] ウィンドウで簡単なクエリを直接作成するか、[Query Designer] アイコンをクリックします。詳細については、「Query Designer」(59 ページ) を参照してください。

6 プロパティ・ウィンドウで、その他のオプション・プロパティおよびデータベース・オプションを指定します。

❖ **昇順インデックス値の再設定**

リポジトリ内のロード・インデックスの永続的な値を直接操作することはできません。ただし、その値を、[Load Index Value] プロパティで保存した値に再設定することができます。実行プロパティを再設定するには、次の操作を実行します。

1 ナビゲータのプロジェクトを右クリックします。

2 [Reset Execution Properties] を選択します。

❖ **ポート構造の更新**

データベースが変更されたポート構造を更新するには、次の手順に従います。

1　DB DataProvider Index Load コンポーネントを右クリックします。

2　[Reconfigure] を選択します。

[Reconfigure] オプションを選択すると、データベース・スキーマが変更されている場合、コンポーネントの設定が更新されます。現在の接続が閉じ、データベースへの新しい接続が開かれ、クエリのメタデータが読み取られ、ポート構造の更新が適用されます。

DataProvider Index Load プロパティ・リスト

DB Data Provider Index Load プロパティ・リストは、[Database Configuration] ウィンドウで定義される接続パラメータおよび他のアイテムを識別します。必須のプロパティは、**太字**テキストで表示されます。

必須プロパティ

プロパティ	説明
[Interface]	データ・ソースへの接続に使用するメソッドまたはドライバを識別します。
[Host Name]	データ・ソースを識別します。ホスト名リストにあるオプションは、選択したインタフェースによって異なります。 SQLite をホストとして使用する場合は、SQLite データベース・ファイルのパスおよびファイル名を入力します (たとえば、*c:¥mySQLite.db*)。指定した名前を持つデータベースが存在しない場合は、SQLite によって作成されます。

プロパティ	説明

[Query]	[Query] アイコンをクリックすると、データ・ソースから情報を取得するクエリを作成できるウィンドウが開きます。WHERE 句での選択基準には、事前に定義された変数 *LoadIndex* による修飾が必要です。クエリはデータベースに送信する前に評価されるため、LoadIndex は次のように角カッコで囲んでください。
	`select * FROM SALES`
	`WHERE SA_DELIVERYDATE >'[LoadIndex]'`
	`ORDER BY SA_DELIVERYDATE`
	注意 引用符文字はデータベース・システム間で異なります。Microsoft Access データベースでは、日付時刻に # を使用します。
	[Query] ウィンドウを使用して簡単なクエリを作成するか、[Query Designer] アイコンをクリックします。詳細については、「Query Designer」(59 ページ) を参照してください。

[Ascending Index]	差分ロードの昇順インデックスが含まれている属性を選択するウィンドウを開きます。

オプション・プロパティ

プロパティ	説明
[User and Password]	認証されたデータベース・ユーザを識別する組み合わせに使用され、認証されていないアクセスからデータベースを保護します。

[Load Index Value]	Sybase ETL ジョブまたはスケジュールを実行するときに、昇順インデックス属性の最大値が自動的に使用され格納されます。[Load Index Value] を使用すると、ユーザが指定した値でプロジェクトをシミュレーションします。例を示します。
	`2005-01-19`
	`100`

[Read Block Size]	1 つのステップでコンポーネントが取得するレコードの数を決定します。

プロパティ	説明
[Pre-processing SQL]	[Pre-processing SQL] アイコンをクリックして、コンポーネントの初期化中に実行されるクエリを作成できるウィンドウを開きます。 クエリには 1 つまたは複数の SQL 文を含めることができます。複数の文を使用する場合は、セミコロン (;) で区切ります。
[Post-Processing SQL]	[Post-processing SQL] アイコンをクリックし、すべてのコンポーネントの実行後に実行されるクエリを作成できるウィンドウを開きます。 クエリには 1 つまたは複数の SQL 文を含めることができます。複数の文を使用する場合は、セミコロン (;) で区切ります。
[Database]	データ・ソースとして使用するデータベースを識別します。 このオプションを選択すると、適切なインタフェース、場合によっては適切なユーザ ID およびパスワードを選択する必要があります。
[Schema]	データ・ソースとして使用するスキーマ／所有者を識別します。表示されたオブジェクトは適切に制限され、新しいテーブルがそのスキーマに作成されます。
[Standardize Data Format]	異なるフォーマットをサポートしているシステム間で Sybase ETL が移動できるように、受信する日付および番号情報を標準フォーマットに変換します。 日付は、年、月、日、時、分、秒、秒の小数点以下が含まれるフォーマット `YYYY-MM-DD hh:mm:ss.s` に変換されます。例を示します。 `2005-12-01 16:40:59.123` 数値は、小数点として「.」を使用して変換されます。
[Database Options]	[Database Options] アイコンをクリックすると、パフォーマンスのデフォルトを上書きし、一部のトランザクションの動作を制御するオプションを設定できるウィンドウが開きます。 詳細については、「データベース接続の設定」(105 ページ) を参照してください。

Index Load のシミュレーション

Index Load コンポーネントが含まれているプロジェクトがすべて設定されていると、これらの手順を実行してインクリメンタル・ロードをシミュレーションすることができます。

1 プロジェクトを非対話型の方法で実行するには、[Run] – [Noninteractive Trace] を選択します。[Load Index Value] プロパティで指定した条件に一致するレコードがすべて処理されます。

2 プロジェクトを対話型の方法でシミュレーションします。Index Load コンポーネントはレコードを 1 件も返しません。詳細については、「プロジェクトのシミュレーション」(31 ページ) を参照してください。

注意 変更されたレコードが正しく取得されているかどうかを確認するには、手順 1 ～ 2 でソース・テーブルを手動で更新します。

3 再びシミュレーションするには、コンポーネントを右クリックして [Reset Load Index Value] を選択するか、プロパティ・ウィンドウで [Load Index Value] プロパティの新しい値を入力します。

DataProvider Index Load のデモ

Sybase ETL には、Data Provider Index Load コンポーネントのデモが複数含まれています。これらのデモは、[Help] メニューの Flash デモ、および DemoRepository のサンプル・プロジェクトとして使用できます。

Flash デモ

Flash デモを実行するには、[Help] - [Demonstrations] - [Source] をクリックして、[DB Data Provider - Index Load] を選択します。

DemoRepository

ナビゲータで、[Repository] - [TRANSFORMER.transformer.Repository] - [Projects] をクリックして、[Demo Transfer U.S. Sales on an incremental basis] を選択します。

Text Data Provider

Text Data Provider は、テキスト・ファイルからテーブルに構造化データを読み取って変換します。テキスト・ソースには固定長または区切られるフィールドが含まれる必要があります。

Text Data Provider コンポーネントの設定

1　Text Data Provider を設計ウィンドウにドラッグします。

2　[Text Data] コンポーネント・ウィンドウで、データ・ソースとして使用するテキスト・ファイルを選択します。

3　プロパティ・ウィンドウで、必要に応じてファイルの説明プロパティを変更します。

特定のフィールドの要件については、「Text Data Provider プロパティ・リスト」(118 ページ) を参照してください。

Text Data Provider コンポーネント・ウィンドウ

Text Data Provider コンポーネント・ウィンドウを使用すると、OUT ポートでデータの構造プロパティを定義できます。このファイルは以下のパネルで構成されます。

- [File Content] ウィンドウ枠 – ソース・ドキュメントの内容を表示します。
- [Properties] ウィンドウ枠 – ファイルの説明プロパティを表示します。
- [Output Port Content] ウィンドウ枠 – OUT ポートでデータのテーブル・ビューを表示します。

Text Data Provider プロパティ・リスト

Text Data Provider プロパティ・リストは、ソース・ファイルの構造に関する項目を識別します。プロジェクトにコンポーネントを追加すると、最初にプロパティが設定されます。必須のプロパティは、**太字**テキストで表示されます。

必須プロパティ

プロパティ	説明
[Text Source]	データ・ソースとして使用するテキスト・ファイルを識別します。プロジェクトに Text Data Provider を追加するか、プロパティ・ウィンドウから Text Data Provider を追加するときにデータ・ソースを選択することができます。プロパティ・ウィンドウでデータ・ソースを選択するには、[Text Source] アイコンをクリックして、ファイルを選択します。
[Columns]	[Columns] アイコンをクリックして、ソース・ファイルにデータのカラムを定義できるプロパティ・シートを開きます。

オプション・プロパティ

プロパティ	説明
[Row Delimiter]	各ローの区切り方を選択します。 • [Position] (固定行位置) • [LF] (改行) • [CR] (行頭復帰) • [CRLF] (行頭復帰とそれに続く改行) 別のデリミタ文字を使用することもできます。
[Row Length]	ロー・デリミタとして [Position] を選択した場合、各固定ローの文字数を指定します。
[Column Delimiter]	カラムの区切り方を選択します。 • [Position] (固定カラム位置) • [Tab] • [Comma] • [Semicolon] 別のデリミタ文字を使用することもできます。

プロパティ	説明
[Column Quote]	次のように、ソース・ファイルの値に引用符を付ける方法を指定します。 • [None] • [Single quote] • [Double quote]
[Fixed by Bytes]	行の長さ、カラムの始まりとカラムの終わりに指定された値を解釈する方法を指定します。 • オフ – 値は、文字数として解釈されます。これがデフォルトです。 • オン – 値は、バイト数として解釈されます。 例 – 次のような特性のソース・ファイルに **abcÖÐÎÄ abcdef** が含まれているとします。 • ファイル・タイプ – 固定長 (可変行) • エンコーディング – GB2312 • ロー・デリミタ – '¥n' • カラム定義 – column1: 1-7; column 2: 9-10 [Fixed by Bytes] オプションを選択した場合 : • カラム 1 には、最初の 7 バイト (**abcÖÐÎÄ**) が表示されます。 • カラム 2 には、9 番目のバイトと 10 番目のバイト (**ab**) が表示されます。 [Fixed by Bytes] オプションを選択しなかった場合: • カラム 1 には、最初の 7 文字 (**abcÖÐÎÄ a**) が表示されます。 • カラム 2 には、それ以降の 2 文字 (**cd**) が表示されます。
[Null Byte Substitute]	null バイトを置換する文字を設定します。
[Skip Rows]	ロー・シーケンスのローの指定した数をスキップします。
[Encoding]	現在の文字コードを設定します。文字コードを設定するには、ドロップダウン・メニューから適切な値を選択します。
[Support Unicode]	Unicode サポートを設定します。
[Read Block Size]	1 つのステップでコンポーネントが取得するレコードの数を決定します。

Text Data Provider のデモ

Sybase ETL には、Text Data Provider コンポーネントのデモが複数含まれています。これらのデモは、[Help] メニューの Flash デモ、および DemoRepository のサンプル・プロジェクトとして使用できます。

Flash デモ

Flash デモを実行するには、[Help] - [Demonstrations] - [Source] をクリックして、[Text Data Provider] を選択します。

DemoRepository

ナビゲータで、[Repository] - [TRANSFORMER.transformer.Repository] - [Projects] をクリックして、次を選択します。

- [Demo Transfer German Customers]
- [Demo Transfer German Sales]
- [Demo Transfer U.S. Customers]

XML via SQL Data Provider

XML via SQL Data Provider は、リレーショナル・データベースと同様にクエリを実行できるリレーショナル・スキーマに階層 XML データをロードします。

XML via SQL Data Provider は、注文、株価、科学データなど、データ中心の XML ドキュメント用に設計されています。これは、通常の階層構造を特徴としています。

XML via SQL Data Provider コンポーネントの設定

1 XML via SQL Data Provider コンポーネントを設計ウィンドウにドラッグします。

2 プロパティ・ウィンドウで、[XML Source] アイコンをクリックして、データ・ソースとして使用する XML ファイルを選択します。*HTTP*、*FTP*、*URL*、ファイル名のいずれかを指定できます。

3　[Data Output] アイコンをクリックします。

4　プロパティ・アイコンをクリックして、XML Port Manager を開きます。各出力ポートのクエリを指定します。

XML via SQL Data にはデフォルトで 1 つの出力ポートが含まれていますが、ポートを追加することができます。XML Port Manager で追加した OUT ポートは設計ウィンドウに表示されます。詳細については、「XML Port Manager の操作」(121 ページ) を参照してください。

❖ **出力ポート構造の更新**

出力ポート構造を更新して XML ソース・ファイルの変更を反映させるには、次の手順に従います。

1　XML via SQL Data Provider コンポーネントを右クリックします。

2　[Reconfigure] を選択します。

[Reconfigure] オプションを選択すると、データベース・スキーマが変更されている場合、コンポーネントの設定が更新されます。現在の接続が閉じ、データベースへの新しい接続が開き、XML ソース・ファイルが読み取られ、出力ポート構造の更新が適用されます。

XML Port Manager の操作

[XML Port Manager] ウィンドウを使用すると、XML ソース・ファイルに対するクエリを作成して、1 つまたは複数の出力データ・ストリームを定義できます。

- [XML Source] ビュー – ソース・ドキュメントの内容を表示します。
- [Data Model] タブ – ソース・ドキュメントのリレーショナル・ビューを表示します。
- [Reference] タブ – 使用可能なコンポーネント変数を表示します。
- [OUT-Port] 領域 – データ・モデルに対するクエリを作成し、特定の OUT ポートに結果を送信するために使用されます。XML Port Manager はデフォルトで 1 つの OUT ポートを伴って設定されますが、別のポートを追加して別のクエリを作成することもできます。

クエリの作成

XML Port Manager を開くと、[OUT-Port] 領域には XML ビューに対する通常のクエリが含まれていて、すべてのカラムおよびローが OUT1 に返されます。XML ソース・ドキュメントには各データ・ノードの属性値として表される顧客データが含まれていることを前提としています。

```
<root>
   <data id="101" fname="Michaels" lname="Devlin"
      address="114 Pioneer Avenue" city="Kingston"
      state="NJ" zip="07070"/>
   <data id="102" fname="Beth" lname="Reiser"
      address="33 Whippany Road" city="Rockwood"
      state="NY" zip="10154"/>
    <data id="103" fname="Erin" lname="Niedringhaus"
      address="190 Windsor Street" city="Tara"
      state="PA" zip="19301"/>
</root>
```

XML に対する顧客属性を取得するには、次と同様のクエリを使用できます。

```
select * from V_XML_CONTENT WHERE TAB_data_ATT_city =
'Kingston'
```

このクエリは Content Browser を開き、city 値が Kingston と一致するローのみを返します。テーブル・ビューでは、次と同様のクエリを作成できます。

```
select * from TAB_data where ATT_city='Kingston'
```

注意 XML データ関係は通常、親／子またはノード／属性の関係として表されます。Content Browser はカラムおよびローとして XML Port Manager クエリ結果を返します。カラムおよびローは、XML データでなくクエリ結果を参照します。

❖ **XML データ・ソースからのデータの取得**

- 次のように、標準 SQL 構文を使用して、[OUT-Port] 領域のポート・フィールドに直接クエリを作成します。

```
select column
FROM table_name
```

- Query Designer を使用すると、テーブル・ビューのクエリを設計できます。XML ソースの構造によっては、テーブル間のジョインを作成してデータのローを返す必要がある場合があります。
- デフォルトの XML view_name は `V_XML_CONTENT` です。ローを返すには、WHERE 句を使用して select 文を修飾します (`select * from V_XML_CONTENT WHERE TAB_state_ATT_state = 'NY'`)。
- テーブル・ビューでは、属性式としてフォーマットされた XML ノードが、ローを返すために WHERE 句で修飾できるラッパ要素を作成することがあります (`select * from TAB_data where ATT_city='Kingston'`)。

❖ **テーブル・ビューに対するクエリの作成**

- [OUT-Port]　領域のポート・フィールドにテーブル・ビューに対するクエリを直接作成するか、Query Designer を使用することができます。テーブル・ビューのテーブルに対してクエリを実行するには、標準 SQL 構文を使用します。

❖ **ポートの追加および削除**

- ポート・セクションを右クリックして、[Add port] または [Remove port] を選択します。

  [Info Port] を追加して、XML ドキュメントを次のコンポーネントに転送します。このポートは、XML Port Manager を終了した後に表示されます。

サンプル・プロジェクトの設定

この項では、簡単な例を使用してコンポーネントの設定方法について説明します。この例に従うには、XML ソースとして *PRODUCTS.xml* を使用します。そのファイルは、Sybase ETL インストール・ディレクトリの Demodata サブディレクトリにあります。

XML Port Manager

XML Port Manager を開いて、[OUT-port] 領域のポートを定義します。各ポートは、XML Data Model テーブルに基づいて select 文で記述されています。

XML ソース

次の XML ドキュメントは簡単な製品構造です。各製品は ID (PR_ID)、名前 (PR_NAME)、製品グループ (PR_GROUP1)、および価格 (PR_PRICE) で記述されています。次に例を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
   <dataroot xmlns:od="urn:schemas-solonde-com:demodata"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="PRODUCTS.xsd"
            generated="2005-01-24T16:13:26"><PRODUCTS>
      <PR_ID>435672</PR_ID>
      <PR_NAME>24 CD Rom Drive</PR_NAME>
      <PR_GROUP1>CD Rom</PR_GROUP1>
      <PR_PRICE>134</PR_PRICE>
   </PRODUCTS>
   <PRODUCTS>
      <PR_ID>435673</PR_ID>
      <PR_NAME>Notebook 235</PR_NAME>
      <PR_GROUP1>Notebook</PR_GROUP1>
      <PR_PRICE>1455</PR_PRICE>
   </PRODUCTS>
</dataroot>
```

データ・モデル

ルート要素にはテーブルが 1 つあり (TAB_dataroot)、その後にレベル 1 の要素に対して 1 つまたは複数のテーブルが続きます。例では、このレベルの要素が 1 つだけ存在します (TAB_PRODUCTS)。次のレベルでは、レベル 2 の各要素に対してテーブルが 1 つあります (TAB_PR_ID、TAB_PR_NAME、TAB_PR_GROUP1、TAB_PR_PRICE)。XML ドキュメントにはネストされたレベルが多く存在し、各レベルが別のテーブルのセットを作成します。

注意 [DB Schema Options] プロパティで生成したテーブル名のプレフィクスを変更することができます。

ルート・レベル	要素レベル 1	要素レベル 2
TAB_dataroot ATT_ROW_ID ATT_FK_generated ATT_xmlns_od ATT_xsi_no	TAB_PRODUCTS ATT_ROW_ID ATT_FK_dataroot	TAB_PR_ID ATT_ROW_ID ATT_FK_PRODUCTS ATT_PR_ID TAB_PR_NAME ATT_ROW_ID ATT_FK_PRODUCTS ATT_PR_NAME TAB_PR_GROUP1 ATT_ROW_ID ATT_FK_PRODUCTS ATT_PR_GROUP1 TAB_PR_PRICE ATT_ROW_ID ATT_FK_PRODUCTS ATT_PR_PRICE

テーブルは外部キーを使用してリンクされます。テーブル TAB_PRODUCTS は、ATT_FK_dataroot 属性を通じて TAB_dataroot にリンクされます。

レベル 2 のテーブルは、ATT_FK_PRODUCTS 属性を通じてテーブル PRODUCTS にリンクされます。PRODUCTS レコードが含まれるビューを作成するには、レベル 2 のテーブルを TAB_PRODUCTS テーブルにジョインさせる必要があります。

ジョインを各レベル 2 のテーブルの ATT_FK_PRODUCTS 属性と TAB_PRODUCTS の ATT_ROW_ID で修飾します。選択した属性だけがレベル 2 のテーブル ATT_PR_ID、ATT_PR_NAME、ATT_PR_GROUP1、および ATT_PR_PRICE の値属性です。

```
select  TAB_PR_ID.ATT_PR_ID,
TAB_PR_NAME.ATT_PR_NAME, TAB_PR_GROUP1.ATT_PR_GROUP1,
TAB_PR_PRICE.ATT_PR_PRICE
FROM  TAB_PRODUCTS, TAB_PR_ID, TAB_PR_NAME,
TAB_PR_GROUP1, TAB_PR_PRICE
WHERE  TAB_PR_ID.ATT_FK_PRODUCTS =
TAB_PRODUCTS.ATT_ROW_ID AND
TAB_PR_NAME.ATT_FK_PRODUCTS = TAB_PRODUCTS.ATT_ROW_ID
AND  TAB_PR_GROUP1.ATT_FK_PRODUCTS =
TAB_PRODUCTS.ATT_ROW_ID AND
TAB_PR_PRICE.ATT_FK_PRODUCTS =
TAB_PRODUCTS.ATT_ROW_ID
```

XML via SQL Data Provider プロパティ・リスト

XML via SQL Data Provider プロパティ・リストは、XML ソース・ファイルの処理オプションを設定します。必須のプロパティは、**太字**テキストで表示されます。

必須プロパティ

プロパティ	説明
[XML Source]	データ・ソースを識別します。 プロジェクトにコンポーネントを追加するか、プロパティ・ウィンドウからファイルを選択するときに、XML データ・ソースを選択できます。プロパティ・ウィンドウでデータ・ソースを選択するには、*[XML Source]* をクリックして、ファイルを選択します。
[Data Output]	[Data Output] アイコンをクリックして、XML Port Manager を開きます。XML Port Manager は XML ソースに対してクエリを実行できる管理コンソールです。 詳細については、「XML Port Manager の操作」(121 ページ) を参照してください。

オプション・プロパティ

プロパティ	説明
[Document Schema]	[Document Schema] アイコンをクリックすると、XML ソースの検証に使用できる外部スキーマ (*.xsd*) または DTD を特定します。
[Namespace Schema]	外部ネームスペース・スキーマのロケーションを示します。XML スキーマには型定義や要素宣言などのコンポーネントで構成されています。これらを使用して、正しい形式の要素および属性情報項目の妥当性を評価します。
[Validate Schema]	スキーマおよび DTD の検証を有効にする場合にこのオプションを選択します。
[XML Options]	[XML Options] をクリックして、次の XML 処理オプションを設定できるウィンドウを開きます。 • [Full schema check] – 時間がかかる項目やメモリを多用する項目をチェックする場合は、この項目を 1 に設定します。部分的ユニーク属性制約チェックと部分的派生制限チェックがこのオプションで制御されます。デフォルト値は 0 です。 • [Ignore external DTD] – ドキュメント内で参照される外部 DTD を無視する場合は、この値を 1 に設定します。デフォルト値は 0 です。 • [Process namespace] – 解析時にネームスペース指定を行わないようにする場合は、この値を 0 に設定します。デフォルト値は 1 です。 • [Preserve Element whitespace] – XML 要素値の空白文字を保持する場合は、この値を true (1) に設定します。false (0) に設定すると、XML 要素値を囲んでいる空白文字が削除されます。XML 要素値が空白文字のみの場合は、空の要素値であると解釈されます。
[DB Schema]	[DB Schema] アイコンをクリックすると、データベース・スキーマ設定 (テーブルの作成) スクリプトが選択されます。このオプションを使用すると、固定データ・モデルが適用されます。

プロパティ	説明
[DB Schema Options]	[DB Schema Options] アイコンをクリックすると、テーブルおよび属性名のプレフィクスを含む、XML 構造から生成したテーブルおよび属性の設定をカスタマイズできるウィンドウが開きます。[DB Schema] オプションは、次のとおりです。 • [Attribute name case] – xml から生成された属性名をフォーマットします。「upper」および「lower」の場合、これに応じて属性名が変換されます。「mixed」の場合、属性名は xml ドキュメントの表示と同じになります。デフォルト値は mixed です。 • [Attribute name prefix] – 生成された属性名ごとにプレフィクスが使用されます。 • [Create indexes] – 1 に設定すると、テーブルのプライマリ・キーのインデックスが自動的に生成されます。デフォルト値は 1 です。 • [Create flat views] – 1 に設定すると、V_XML_CONTENT という名前のビューが自動的に生成されます。このビューによって、すべてのテーブルが結合され、すべての xml データが 1 つのまとまったテーブルで返されます。デフォルト値は 1 です。 データベース・スキーマのテーブル数が 32 以上の場合、このビューは機能しないため、このオプションをオフに変更する必要があります。 • [Foreign key prefix] – 外部キーである属性にプレフィクスが使用されます。 • [Ignore Empty Leaf Element Values] – データが含まれていない特定の XML リーフ要素のカラム・エントリをデータベースに含めない場合、この値を true (1) に設定します。false (0) に設定すると、空かどうかにかかわらず、XML ドキュメントから作成されたデータベースにすべての XML リーフ要素のカラム・エントリが含まれます。 • [Primary key name] – プライマリ・キーに属性名が使用されます。 • [Table name case] – xml から生成されたテーブル名をフォーマットします。「upper」および「lower」の場合、これに応じてテーブル名が変換されます。「mixed」の場合、テーブル名は xml ドキュメントの表示と同じになります。 • [Table name prefix] – 生成されたテーブル名にプレフィクスが使用されます。

プロパティ	説明
[Database Options]	[Database Options] アイコンをクリックすると、パフォーマンスのデフォルトを上書きし、一部のトランザクションの動作を制御するオプションを設定できるウィンドウが開きます。 詳細については、「データベース接続の設定」(105 ページ) を参照してください。
[Read Block Size]	1 つのステップでコンポーネントが取得するレコードの数を決定します。コンポーネントに 2 つ以上の出力ポートがある場合、読み込みブロック・サイズは無視され、コンポーネントは 1 つのステップですべてのポートのデータを指定します。

XML via SQL Data Provider のデモ

Sybase ETL には、XML via SQL Data Provider コンポーネントのデモが複数含まれています。これらのデモは、[Help] メニューの Flash デモ、および DemoRepository のサンプル・プロジェクトとして使用できます。

Flash デモ

Flash デモを実行するには、[Help] - [Demonstrations] - [Source] をクリックして、[XML via SQL - Data Provider] を選択します。

DemoRepository

ナビゲータで、[Repository] - [TRANSFORMER.transformer.Repository] - [Projects] をクリックして、[Demo XML via SQL Data Provider] を選択します。

Transformation コンポーネント

Transformation コンポーネントには、IN ポートと OUT ポートが少なくとも 1 つずつあり、変換ストリームのデータに特定の変換が適用されます。

コンポーネント	説明
Data Calculator JavaScript	このコンポーネントに渡されたすべてのレコードに対して IN ポートと OUT ポートの間で変換を実行します。
Data Splitter JavaScript	入力値に基づいて受信データ・ストリームを分割します。
Character Mapper	入力レコードの文字および文字列を置き換えます。Character Mapper は、選択されたすべての属性に置換マッピングを適用します。

Data Calculator JavaScript

Data Calculator JavaScript は、Transformation コンポーネントの 1 つです。このコンポーネントは、渡されたレコードに対して IN ポートと OUT ポートの間で変換を適用する規則を定義するために使用します。たとえば、Data Calculator JavaScript を使用してポート属性を変換する規則を定義したり、新しい属性を作成する規則を追加したりするために使用できます。

Data Calculator では、属性レベルの検索を実行することもできます。特定の検索ポートに検索データを入力する必要があります。

シミュレーション・シーケンスへの影響は次のとおりです。

検索なしでは、Data Calculator JavaScript はシミュレーション・シーケンスに影響を与えません。検索を使用すると、メイン・ポートのデータが処理される前にすべてのデータが検索ポートに読み込まれます。

Data Calculator JavaScript コンポーネントの設定

1　Data Calculator JavaScript コンポーネントを設計ウィンドウにドラッグします。

2　必要に応じて、Data Calculator の IN ポートを、インバウンド・データを提供するコンポーネントの OUT ポートにリンクします。

3　Data Calculator コンポーネント・ウィンドウが開いている場合は、[Save] をクリックしてウィンドウを閉じます。

4　設計ウィンドウで Data Calculator の OUT ポートを右クリックし、[Assign Structure] を選択してから、次のオプションのいずれかを選択します。

オプション	対処法
[IN]	Data Calculator の IN ポート構造と一致する OUT ポート構造を作成します。
[Copy Structure]	既存のポート構造を OUT ポートに適用するためのウィンドウを開きます。

5　設計ウィンドウで [Data Calculator JavaScript] をダブルクリックするか、プロパティ・ウィンドウの [Rules] アイコンをクリックします。コンポーネント・ウィンドウが開き、デフォルト・マッピングを順番に割り当てるよう求めるプロンプトが表示されます。

6　次のいずれかのオプションを選択します。

オプション	対処法
[Yes]	デフォルト・マッピングを順番に作成します。 IN ポートの各属性を対応する OUT ポートの属性にマッピングする一連の変換規則を作成するには、このオプションを選択します。
[No]	IN ポートから OUT ポートへの独自のマッピングを定義します。 IN ポートの属性を対応する OUT ポートの属性に個別にマッピングするには、このオプションを選択します。詳細については、「ポート属性のマッピング」(133 ページ) を参照してください。

マッピング・オプションと変換規則の詳細については、「コンポーネント・ウィンドウの操作」(132 ページ) を参照してください。

コンポーネント・ウィンドウの操作

Data Calculator コンポーネント・ウィンドウでは、データ・ストリームのテーブル・ビューとグラフィック・ビューが表示されます。このウィンドウは、ポート属性をマッピングし、変換規則を定義するときにも使用できます。

テーブル・ビュー

テーブル・ビューでは、現在のレコード構造、ポート属性、および変換規則の構造的なビューが表示されます。このファイルは以下のパネルで構成されます。

- [Current Input Record] ウィンドウ枠は、IN ポートの現在のレコードに含まれる各属性および属性値を示します。すべての IN ポート属性に IN. プレフィクスが含まれます。
- [Transformation rules and Current Output Record] ウィンドウ枠には、変換規則ごとにカラムとロー、および OUT ポート属性ごとにカラムとローが含まれます。デフォルト変換規則は、各 IN ポート属性を対応する OUT ポート属性にマッピングします。デフォルトでは OUT ポート値は IN ポート属性を反映します。属性値または変換規則を変更すると、OUT ポート値も変更されます。

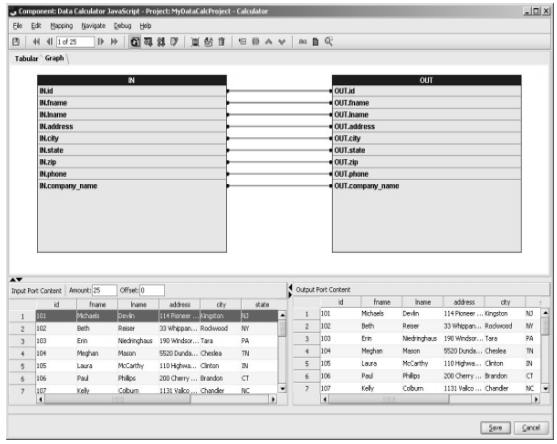
  変換規則が関数式である場合、1 行に入力する必要があります。デフォルトでは、戻り値は、対応する OUT ポート属性に割り当てられます。ただし、複数行の関数式を入力した場合は、テキストはスクリプトとして解釈されるので、ユーザが OUT ポート属性値を設定する必要があります。
- [Input Port Content] 領域には、IN ポートで現在利用可能な一連のレコードが表示されます。
- [Output Port Content] 領域には、OUT ポートで現在利用可能な一連のレコードが表示されます。

グラフ・ビュー

グラフ・ビューには、IN ポート属性と OUT ポート属性間の現在のマッピングが表示されます。

- [Input Port Content] 領域には、IN ポートで現在利用可能な一連のレコードが表示されます。
- [Output Port Content] 領域には、OUT ポートで現在利用可能な一連のレコードが表示されます。

注意 変換規則を IN.attribute に適用すると、IN.attribute と OUT.attribute 間のマッピング・ラインは表示されなくなります。



❖ ポート属性のマッピング

Data Calculator ではデフォルト・オプションとして IN ポートと OUT ポート間のカラム単位のマッピングが作成されますが、ポート属性を個別にマッピングすることが必要になる場合もあります。独自のマッピングを作成するには、グラフィック・ビューを使用します。

1　Data Calculator コンポーネント・ウィンドウで、[Graph] タブをクリックします。

2 次のいずれかの方法で IN ポート構造を OUT ポート構造にマッピングします。

- メニュー・バーで [Mapping] を選択し、次の定義済みのマッピング・シーケンスの 1 つを選択します。
  - [Create mapping by Order] – IN 構造と OUT 構造のポート属性を順番にマッピングします。

    注意 双方の属性数が異なる場合、一部のポート属性はマッピングされません。

  - [Create mapping by Name] – 名前に従って IN 構造と OUT 構造のポート属性をマッピングします。
  - [Create mapping by Name Case Sensitive] – 大文字と小文字を区別した名前に従って IN 構造と OUT 構造のポート属性をマッピングします。
  - [Create mapping by prefix] – 指定されたプレフィクスは無視し、名前に従って IN 構造と OUT 構造のポート属性をマッピングします。
  - [Create mapping by Best Match] – よく似た IN 構造と OUT 構造のポート属性をマッピングします。
- IN ポート属性と OUT ポート属性を個別に接続します。

コンポーネントが使用できる状態になりました。IN ポートから OUT ポートにレコードを転送できます。

変換結果の表示

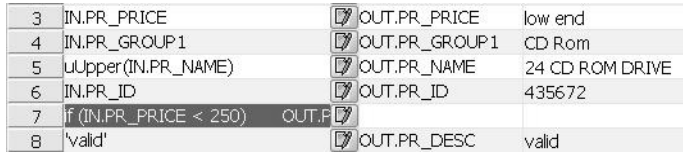
Data Calculator に不可欠な機能の 1 つに、変換規則の結果を即時表示する機能があります。このシミュレーション機能を使用すると、受信データの検証、変換結果のテスト、およびデータ出力における規則の効果を確認できます。

現在の入力レコード属性値

デフォルトでは、出力ポートの値に入力ポートの値が反映されます。入力ポートの属性値を変更すると、変換規則をテストできるほか、現在の出力レコードでその結果を確認できます。この領域に手動で加えた変更は、入力または出力ポート・バッファのみに反映されます。これは、変換規則のテスト・ケースを作成するために便利な方法です。

変換規則の編集

[Transformation Rule] カラムでは、変換規則の追加や変更、または削除が可能です。さらに、現在の属性入力フィールドを変更することにより、単一行の関数を編集できます。関数 uUpper を属性 IN.PR_NAME に追加する場合は、次の例のようになります。



プロシージャ・エディタを使用できます。複雑な手続き型変換の詳細については、「JavaScript Editor and Debugger の使用」(75 ページ) を参照してください。

また、変換規則を追加することもできます。これは、OUT 属性の数が IN 属性の数と一致しない場合や、開発の後の段階でプロジェクトに別の属性を追加する場合に非常に便利です。

注意 グラフィック・ビューは、実際のポート構造を反映します。このビューでは属性を追加したり削除したりできません。

❖ 変換規則の管理

1　変換規則を追加するには、[Transformation Rule] カラムまたは [Current Output Port] カラムの任意の場所を右クリックします。

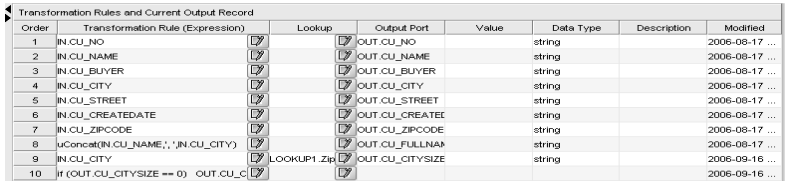
2　[Insert] を選択します。

追加された規則を使用して、追加の割り当てや計算を行うことが可能になります。

- 変換規則を削除するには、[Transformation Rule] カラム内のローを右クリックし、[Remove] を選択します。
- 変換規則の順序を変更するには、[Transformation Rule] カラム内のローを右クリックし、[Up] または [Down] を選択します。
- 欠落している出力属性を追加するには、[Mapping] をクリックし、[Add missing output attributes] を選択します。

変換規則の処理の順序

変換規則は、順番に処理されます。処理は、リストの最初の変換規則から開始されます。次の例を考えます。



Transformation Rules and Current Output Record

Order	Transformation Rule (Expression)	Lookup	Output Port	Value	Data Type	Description	Modified
1	IN.CU_NO		OUT.CU_NO		string		2006-08-17 ...
2	IN.CU_NAME		OUT.CU_NAME		string		2006-08-17 ...
3	IN.CU_BUYER		OUT.CU_BUYER		string		2006-08-17 ...
4	IN.CU_CITY		OUT.CU_CITY		string		2006-08-17 ...
5	IN.CU_STREET		OUT.CU_STREET		string		2006-08-17 ...
6	IN.CU_CREATEDATE		OUT.CU_CREATEI		string		2006-08-17 ...
7	IN.CU_ZIPCODE		OUT.CU_ZIPCODE		string		2006-08-17 ...
8	uConcat(IN.CU_NAME,',',IN.CU_CITY)		OUT.CU_FULLNAI		string		2006-08-17 ...
9	IN.CU_CITY	LOOKUP1.Zip	OUT.CU_CITYSIZE		string		2006-09-16 ...
10	if (OUT.CU_CITYSIZE == 0) OUT.CU_C						2006-09-16 ...

行 9 は、市区町村の郵便番号の数を検索し、CitySize に結果を保存します。行 10 のプロシージャは、CitySize の数値に基づいて CitySize 文字列を計算します。

Data Calculator のシミュレーション

Data Calculator は、変換規則の処理でデータに加えられる変更をユーザが確認できるよう設計されています。これは、たとえば変換規則の変更が送信データにどのように影響を与えるかを確認する場合などに便利です。[Auto-Synchronization] ボタンのステータスに応じて、規則の入力後、変換規則はすぐにすべての IN ポート・レコード・セット全体に適用されます。

自動同期の切り替え :

自動同期機能により、変換規則に加えたすべての変更が IN ポートの現在の全レコードに適用されます。

注意 自動同期機能を無効にすると、[Step] オプションを選択することにより、IN ポート・データの処理を手動でトリガできます。

すべての変換規則の手動による適用 :

すべての変換規則を IN ポートの現在のすべてのレコードに手動で適用するには、ツールバーの [Step] アイコンをクリックします。

別のレコード セットのフェッチ :

別のレコード セットをフェッチするには、ツールバーにある [Step through the next incoming data buffer] アイコンをクリックします。

入力ポートレコードのステップ実行 :

IN ポートのレコードをステップ実行するには、次の手順に従います。

- ツールバーで適切なレコード・コントロールをクリックします。
- [Navigate] をクリックし、適切なオプションを選択します。
- [Input Port Content] リストでレコードを選択します。

注意 [Current Input Record] 領域に示された値は、現在のレコードの変更とともに更新されます。

変換規則でのキーワードの検索 :

ツールバーの [Search Content of Transformation Rules] アイコンをクリックします。

null 値および空の値のハイライト :

ツールバーで [Highlight NULL-Values and Empty Values] アイコンをクリックします。

Data Calculator での検索の使用

Data Calculator では、属性レベルの検索を実行します。特定の検索ポートに検索データを入力する必要があります。

検索ポートの追加

Data Calculator に検索ポートを追加するには、データの提供元のコンポーネントの OUT ポートを直接 Data Calculator コンポーネント (ポートではない) に接続します。検索ポートが自動的に作成され接続されます。あるいは、Data Calculator を右クリックして [Add Input Port] を選択し、データの提供元のコンポーネントの OUT ポートを新しいポートに接続します。検索ポートの数の制限はありません。

検索データの準備

各検索ポートには少なくとも 2 つの属性が必要です。最初の属性はキーを表します。その他の属性はすべて戻り値を表します。

複合キーを検索する場合、先行のコンポーネント内のキー値を連結し、検索のキー式でも同じ連結方法を使用する必要があります。

一般的な検索オプションの設定

検索を設定するには [Lookup Options] プロパティを使用します。プロパティ・ウィンドウには、すべての検索ポートおよびそれらの現在のオプション値のリストが表示されます。

- [Lookup Name] — 関連付けられたポートから継承され、ここでは変更できません。ポート名を変更するには、ポート・メニューで [説明] を選択します。
- [Lookup Size] — 予想される検索レコードの数を入力し、メモリの割り付けおよび検索のパフォーマンスを最適化します。詳細については、「DB Lookup」(148 ページ) を参照してください。
- [Lookup Empty / Null] — 空の値と null 値は通常「不定」キーとして処理され、検索のデフォルト値が返されます。空の値または null 値が検索の有効なキーである場合は、このオプションをアクティブ化してそれらのキーの値の検索を実施できます。

検索規則の構築

検索規則を設定するには、[Data Calculator] ウィンドウで [Tabular] タブを開きます。検索ポートが使用可能な場合は、追加の [Lookup] カラムが表示されます。

各検索規則について次の情報を入力します。

- [Key Expression] — 検索リストの最初のカラムで検索する値です。変換規則としてキー式 (IN.PR_ID など) を入力します。
- [Return Value] — 検索リストには、複数の戻り値カラムが含まれる場合があるので、キーが検出されたときに返す値を指定する必要があります。返す値を指定するには、[Lookup] カラムのポップアップ・メニューで関連付けられたポート属性を選択します。次に例を示します。
  LOOKUP1>>LOOKUP1.PR_NAME

注意 戻り値が選択され、名前順に表示されますが、検索では内部でカラム番号を使用します。これは、特に属性を追加したり削除したりした場合など、検索ポート構造が変更されたときは必ず検索規則を確認する必要があることを意味します。

- [Output Variable] — 検索の戻り値はこの変数に割り当てられます。変数は、[Output Port] カラム (たとえば OUT>>OUT.PR_NAME など) のポップアップ・メニューで選択できます。
- [Default Expression] (オプション) — 指定のキー値が見つからない場合、検索は Null を返します。異なるデフォルト値を返すには、[Lookup Options] ウィンドウに式を入力します。[Lookup Options] ウィンドウを開くには、[Transformation Rules and Current Output Record] パネルの [Lookup] カラムにあるアイコンをクリックします。[Lookup Options] ウィンドウは、デフォルトの式の指定に使用します。このウィンドウには、検索属性および出力属性を選択するためのポップアップ・メニューが含まれます。

Data Calculator JavaScript のデモ

Sybase ETL には、Data Calculator コンポーネントのデモが複数含まれています。これらのデモは、[Help] メニューの Flash デモ、および DemoRepository のサンプル・プロジェクトとして使用できます。

Flash デモ

Flash デモを実行するには、[Help] - [Demonstrations] - [Transform] をクリックし、[Data Calculator - JavaScript] を選択します。

DemoRepository

ナビゲータで、[Repository] - [TRANSFORMER.transformer.Repository] - [Projects] をクリックします。

検索なしのデモの場合は、以下を選択します。

- [Demo Transfer U.S. Products]
- [Demo Transfer German Products]
- [Demo Data Calculator]

検索のあるデモの場合は、以下を選択します。

- [Demo Transfer German Customers]
- [Demo Transfer German Sales]
- [Demo Transfer U.S. Customers]

Data Splitter JavaScript

Data Splitter JavaScript では、入力データを簡単にフィルタし配布できます。

Data Splitter JavaScript コンポーネントの設定

1 Data Splitter JavaScript コンポーネントを設計ウィンドウにドラッグします。

2 Data Splitter の IN ポートを、インバウンド・データを提供するコンポーネントの OUT ポートにリンクします。

3 Data Splitter の OUT ポートを、アウトバウンド・データの配信先となるコンポーネントの IN ポートにリンクします。

- OUT1 は上の OUT ポートです。
- OUT2 は下の OUT ポートです。

アウトバウンド・データの配信先となるコンポーネントの IN ポート構造を設定する必要があります。

4 Data Splitter JavaScript コンポーネントをダブルクリックします。

5 次のように、データ・フローの方向付けに使用する条件を追加します。

a 条件を追加するポート (OUT1、OUT2) を右クリックし、[Edit] を選択します。

b [Condition] ウィンドウで、各カラムに適用する条件を定義します。

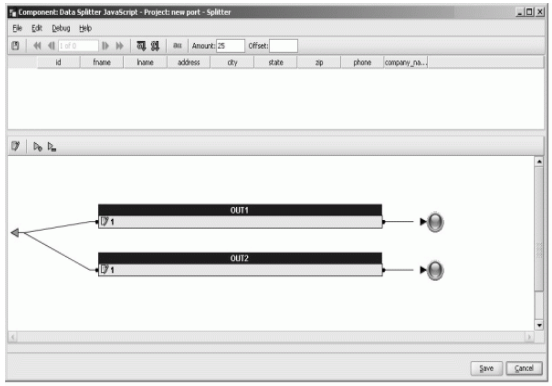
c [Save] をクリックします。

インバウンド・データの分割

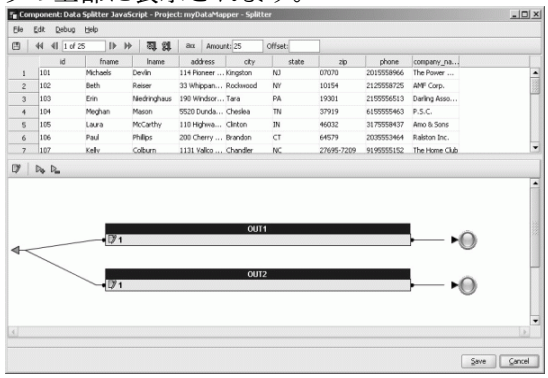
Data Splitter をプロジェクトに追加すると、インバウンド・データ属性および OUT ポート条件を示すコンポーネント・ウィンドウが表示されます。

Data Splitter は、OUT1 と OUT2 の 2 つの OUT ポートを持つように設定できます。OUT1 ポートは上の OUT ポートで、OUT2 ポートは下の OUT ポートです。OUT1 および OUT2 ポートの IN ポート条件は、あらかじめ 1 に設定されています。この条件は常に true であるため、IN ポートの現在の値にかかわらず、Data Splitter はすべての受信レコードを両方の出力ポートにコピーします。

IN ポート データ・バッファは、最初は空であるため、インバウンド・データ属性のみが表示されます。OUT1 および OUT2 の OUT ポート構造は、IN ポート構造と一致します。



入力属性を移植するには、ツールバーの [Step to the next input buffer] アイコンをクリックします。入力データは、コンポーネント・ウィンドウの上部に表示されます。



入力領域と OUT ポートのレコード間には関係があります。OUT1 と OUT2 は IN ポートと同じポート構造を共有するため、上部ウィンドウのレコードを選択すると、OUT ポートはそのレコードがポート条件を満たしているかどうかを示します。レコードがポート条件を満たしている場合は、OUT ポートの色は緑になります。レコードがポート条件を満たしていない場合は、OUT ポートの色は赤になります。

注意 Data Splitter が OUT ポートに転送するレコード数は、受信レコードの数によって異なります。1 つのレコードが複数のポート条件を満たしている場合は、それらのポートのすべてでそれを使用できます。どの条件も満たしていないレコードは、データ・ストリームから削除されます。

ポート条件のカスタマイズ

各ポートに条件を割り当てることができます。条件は、1 つまたは複数の式で構成されます。複数の式は演算子によって連結されます。Data Splitter で条件を評価する場合、結果は TRUE (1) または FALSE (0) になります。

たとえば、OUT1 に対して 1 つの条件セットを作成し、OUT2 に対して別の条件セットを作成することが必要になる場合があります。

❖ ポート条件の変更

1　Data Splitter JavaScript コンポーネントをダブルクリックして、Data Splitter コンポーネント・ウィンドウを開きます。ポートを右クリックして [Edit] を選択します。

2　[Condition] ウィンドウで、ポートに適用する条件を作成します。次のことができます。

- [Condition] ウィンドウのテキスト領域に条件を手動で入力する。
- 条件に追加する変数および関数を左のウィンドウ枠からテキスト領域にドラッグ・アンド・ドロップする。[Variables] タブには使用できるすべての変数が、さらに [Functions] タブには条件に追加できるすべての関数および演算子がリストされます。
- テキスト領域を右クリックし、条件に追加する変数を [IN] ポップアップ・メニューから選択する。

❖ 新しい出力ポートの追加

Data Splitter は、2 つの OUT ポートに設定されていますが、追加の OUT ポートを作成できます。新しいポートは名前で識別します。

1　ポートのツールバーの [Add new port] アイコンをクリックします。

2　新しい出力ポートの名前を入力し、[OK] をクリックします。

❖ 出力ポートの削除

1　削除するポートを選択します。

2　ポート・ツールバーで [Remove selected port] アイコンをクリックします。あるいは、ポートを右クリックして [Delete] をクリックします。

注意 すべてのポートを削除することはできません。コンポーネントには少なくとも 1 つの出力ポートが必要です。

特殊なポート条件

ポート条件により、Data Splitter がレコードを分散させる方法が決定されます。相互排他的ではない重複するポート条件を定義できます。

- 1 — TRUE。同時に他のポート条件とも一致するレコードも含め、すべてのレコードがこのポートに転送されます。
- (空) — Data Splitter で定義されたその他の条件に一致しなかったすべてのレコードが、空の条件のポートに集まります。

Data Splitter JavaScript のデモ

Sybase ETL には、Data Splitter コンポーネントのデモが複数含まれています。これらのデモは、[Help] メニューの Flash デモ、および DemoRepository のサンプル・プロジェクトとして使用できます。

Flash デモ

Flash デモを実行するには、[Help] - [Demonstrations] - [Transform] をクリックし、[Data Splitter - JavaScript] を選択します。

DemoRepository

ナビゲータで、[Repository] - [TRANSFORMER.transformer.Repository] - [Projects] をクリックして、次を選択します。

- [Demo Data Splitter]
- [Demo Text Data Sink Delimited/Fixed]

Character Mapper

Character Mapper は、入力レコード内の文字および文字列を置換する Transformation コンポーネントです。Character Mapper は、選択された属性以外のすべての属性に置換マッピングを適用します。

Character Mapper を使用して文字または文字列を置換します。たとえば、ドイツ語のウムラウト文字 ä を ae または Unicode 文字に置き換えます。

Character Mapper コンポーネントの設定

1　Character Mapper コンポーネントを設計ウィンドウにドラッグします。

2　Character Mapper の IN ポートを、インバウンド・データを提供するコンポーネントの OUT ポートにリンクします。Character Mapper の OUT ポートを、アウトバウンド・データの配信先となるコンポーネントの IN ポートにリンクします。

アウトバウンド・データの配信先となるコンポーネントの IN ポート構造を設定する必要があります。

3　Character Mapper コンポーネント・ウィンドウを開きます。必要に応じて、ツールバーの [Step to next incoming data buffer] ボタンをクリックして入力および出力コンテンツを移植します。

4　マッピング定義を追加します。詳細については、「新しいマッピング定義の作成」(145 ページ) を参照してください。

Character Mapper コンポーネント・ウィンドウの操作

Character Mapper コンポーネント・ウィンドウを使用して、IN ポートと OUT ポート間で渡されるデータのマッピング規則を定義します。

Character Mapper コンポーネント・ウィンドウには以下が含まれます。

- [Current Input Record] ウィンドウ枠。現在 IN ポートにあるレコードのカラム、ロー、およびコンテンツを表示します。
- [Current Output Record] ウィンドウ枠。OUT ポートに表示される現在のレコードのカラム、ロー、およびコンテンツを表示します。
- [Mapping definition] ウィンドウ枠。[From] カラムと [To] カラムを含みます。

❖ 新しいマッピング定義の作成

1　ツールバーの [Insert Mapping] アイコンをクリックするか、[Mapping definition] ウィンドウ枠の任意の場所を右クリックして [Insert Mapping] を選択します。新しいマッピング定義のために新しいローが挿入されます。

2　[From] カラムには置換対象の文字を入力し、[To] カラムには置換後の値を入力します。Character Mapper により規則が適用され、[Current Output Record] ウィンドウ枠に結果が表示されます。

- [Current Input Record] ウィンドウ枠に現在表示されているレコードを編集するには、[Current Input Port Content] ウィンドウ枠内のローをクリックして変更を加えます。[Current Output Record] ウィンドウ枠の値、および [Current Outport Port] ウィンドウ枠で選択されたローの値も更新されます。
- マッピング定義を削除するには、マッピング定義を右クリックし、[Remove Mapping] を選択します。選択したマッピング定義が削除されます。
- マッピング定義の順序を変更するには、ツールバーの [Move row up] アイコンおよび [Move row down] アイコンを選択します。
- 規則が作成されるとすぐに Character Mapper によりマッピングが適用され、出力結果が更新されます。このような自動同期のオン／オフを切り替えるには、ツールバーの [Enable auto refresh of output values] アイコンをクリックします。

注意 Character Mapper ではすべてのカラムとローにマッピングが適用されます。文字マッピングからカラムを除外するには、プロパティ・ウィンドウで [Exclude] をクリックし、除外するカラムを選択します。

❖ マッピング定義の再利用

文字マッピングの定義は、ファイルに保存して、他のプロジェクトで使用することができます。文字マッピングを定義ファイルに保存して、他のプロジェクトで使用するには、次の手順に従います。

- ツールバーの [Export character mapping] アイコンおよび [Import character mapping] アイコンをクリックします。詳細については、「マッピング定義のエクスポート」および「マッピング定義のインポート」を参照してください。

❖ マッピング定義のエクスポート

1 Character Mapper コンポーネント・ウィンドウのツールバーの [Export character mapping] アイコンをクリックします。

2 ファイル名を指定し、[Save] をクリックします。

注意 Character Mapper では、拡張子なしでファイルが保存されます。

❖ **マッピング定義のインポート**

1　Character Mapper コンポーネント・ウィンドウのツールバーの [Import character mapping] アイコンをクリックします。

2　インポートするファイルを選択し、[Open] をクリックします。

Character Mapper のデモ

Sybase ETL には、Character Mapper コンポーネントのデモが複数含まれています。これらのデモは、[Help] メニューの Flash デモ、および DemoRepository のサンプル・プロジェクトとして使用できます。

Flash デモ

Flash デモを実行するには、[Help] - [Demonstrations] - [Transform] をクリックし、[Character Mapper] を選択します。

DemoRepository

ナビゲータで、[Repository] - [TRANSFORMER.transformer.Repository] - [Projects] をクリックして、[Demo Character Mapper] を選択します。

Lookup コンポーネント

検索オペレーションでは、一連のキーと値のペアを格納する検索テーブルのキーに対応する値を検索します。プロジェクトの実行時に静的検索テーブルをキャッシュすることができますが、Lookup をキャッシュせずに動的に実行することもできます。

コンポーネント	説明
DB Lookup	データベース内の検索値。検索データは、必ず検索キーと検索値の 2 つのカラムを返すクエリの結果セットによって指定されます。 検索によって返される値 (検索値) は、現在のレコードの任意の属性に割り当てることができます。検索テーブルは、プロジェクト実行時にキャッシュされます。プロジェクト実行時に、基本となるデータベースに適用された変更は、検索結果には反映されません。

コンポーネント	説明
DB Lookup Dynamic	動的検索を実行するには、クエリの WHERE 句のキー値を参照するクエリを使用します。DB Lookup コンポーネントとは異なり、DB Lookup Dynamic では、検索情報はキャッシュされず、コンポーネントを渡すレコードごとに SQL 検索を 1 回実行します。 プロジェクト実行時に、検索テーブルのデータは、データベースの同時ユーザ (または同じプロジェクト内のユーザ) によって変更されている場合があります。その場合、動的ではないデータベース検索により無効なデータを検索できます。現在の値を見つけるには DB Lookup Dynamic を使用します。

DB Lookup

DB Lookup は、データベース内の値を検索します。検索データは、必ず検索キーと検索値の 2 つのカラムを返すクエリの結果セットによって指定されます。

検索によって返される値 (検索値) は、現在のレコードの任意の属性に割り当てることができます。DB Lookup は、プロジェクトの実行時に検索テーブルをキャッシュします。プロジェクト実行時に、基本となるデータベースに適用された変更は、検索結果には反映されません。

DB Lookup コンポーネントの設定

1　DB Lookup コンポーネントを設計ウィンドウにドラッグします。

2　設計ウィンドウで、DB Lookup の IN ポートをインバウンド・データを提供するコンポーネントの OUT ポートに接続します。

3　プロパティ・ウィンドウで有効なインタフェースとホスト名を指定します。

特定のフィールドの要件については、「DB Lookup プロパティ・リスト」(150 ページ) を参照してください。

4　プロパティ・ウィンドウで、キー属性と値属性を指定します。これらが検索値です。

5　プロパティ・ウィンドウで、[Query] アイコンをクリックして [Query] ウィンドウを開きます。

6　[Query] ウィンドウで検索データを取得するためのクエリを作成し、保存します。ソース・テーブルから検索キーと検索値を返すようにクエリを設計します。

例

ドイツの製品に使用する製品番号を米国で使用する製品番号に置換するとします。ドイツの製品は、テーブル PRODUKTE(PR_NUMMER, PR_NAME, PR_PREIS) にあります。DB Lookup コンポーネントの IN ポートにはこれら 3 つの属性が含まれています。

米国の製品番号の検索を実行するテーブルは、LOOKUP_PRODUCTS(SOURCE, DESTINATION) です。SOURCE カラムにはドイツの製品番号が、DESTINATION カラムには米国の製品番号が含まれます。

ドイツの PR_NUMMER の値が LOOKUP_PRODUCTS に見つからない場合は、現在の PR_NUMMER は「INVALID」という文字列で置換されます。正常に実行された検索では、ドイツの製品番号が対応する米国の番号で置換されます。

この例の DB Lookup コンポーネントを設定するには、以下を選択します。

- [Key Attribute]：PR_NUMMER
- [Value Attribute]：PR_NUMMER
- [Default Value]：INVALID
- [Query]：select SOURCE, DESTINATION FROM LOOKUP_PRODUCTS

DB Lookup プロパティ・リスト

DB Lookup プロパティ・リストは、[Database Configuration] ウィンドウで定義される接続パラメータおよび他のプロパティを識別します。必須のプロパティには、**太字**のラベルが付けられています。

必須プロパティ

プロパティ	説明
[Key Attribute]	IN ポート属性のリストからキー属性を選択します。この属性は、検索テーブルの最初のカラムに対応します。
[Value Attribute]	[Value Attribute] リストから、検索で見つけた値を割り当てる属性を選択します。返された検索値は、既存の値を上書きします。 キー属性と値属性の両方は、レコード構造の同じ属性を参照できます。それにより、対応する値でキーを上書きすることができます。
[Interface]	データ・ソースへの接続に使用するメソッドまたはドライバを指定します。
[Host Name]	データ・ソースを指定します。ホスト名リストにあるオプションは、選択したインタフェースによって異なります。
[Query]	[Query] アイコンをクリックして、データ・ソースから情報を取得するクエリを作成します。[Query] ウィンドウを使用して単純なクエリを作成するか、[Query Designer] アイコンをクリックして Query Designer を開きます。

オプション・プロパティ

プロパティ	説明
[User and Password]	認証されたデータベース・ユーザとパスワードを指定し、認証されていないアクセスからデータベースを保護します。
[Default Value]	値属性に割り当てるデフォルト値を指定します。DB Lookup では、検索テーブルでキー値が見つからない場合にこの値を使用します。
[Use Key Value]	検索が失敗した場合に、デフォルトではなくキー値を値属性に割り当てるには、このオプションを選択します。
[Lookup Empty/Null Keys]	空白または NULL のキー値の検索を実行する場合は、このオプションを選択します。それ以外の場合は、選択したデフォルトの方法が適用されます。

プロパティ	説明
[Lookup Size]	予想される検索レコードの数を指定し、メモリの割り付けおよび検索のパフォーマンスを最適化します。
[Database]	データ・ソースとして使用するデータベースを指定します。さらに、適切なインタフェース、場合によっては適切なユーザ ID およびパスワードを選択する必要があります。
[Schema]	データ・ソースとして使用するスキーマ／所有者を識別します。表示されたオブジェクトは適切に制限され、新しいテーブルがそのスキーマに作成されます。
[Standardize Data Format]	異なるフォーマットをサポートするシステム間で Sybase ETL が移動できるように、受信する DATE および NUMBER 情報を標準フォーマットに変換するオプションを選択します。 日付は、年、月、日、時、分、秒、秒の小数点以下が含まれるフォーマット YYYY-MM-DD hh:mm:ss.s に変換されます。 例を示します。 2005-12-01 16:40:59.123 数値は、小数点として「.」を使用して変換されます。
[Database Options]	[Database Options] アイコンをクリックして、パフォーマンスのデフォルトを上書きし、一部のトランザクションの動作を制御するオプションを設定します。 詳細については、「データベース接続の設定」(105 ページ) を参照してください。

DB Lookup のデモ

Sybase ETL には、DB Lookup コンポーネントのデモが複数含まれています。これらのデモは、[Help] メニューの Flash デモ、および DemoRepository のサンプル・プロジェクトとして使用できます。

Flash デモ

Flash デモを実行するには、[Help] - [Demonstrations] - [Lookup] をクリックし、[DB Lookup] を選択します。

DemoRepository

ナビゲータで、[Repository] - [TRANSFORMER.transformer.Repository] - [Projects] をクリックして、次を選択します。

- [Demo DB Lookup]
- [Demo Transfer German Products]

DB Lookup Dynamic

DB Lookup Dynamic では、クエリの WHERE 句でキー値を参照するクエリを使用して動的検索を実行します。DB Lookup コンポーネントとは異なり、DB Lookup Dynamic では、検索情報はキャッシュされず、コンポーネントを渡すレコードごとに SQL 検索を 1 回実行します。

プロジェクト実行時に、検索テーブルのデータは、データベースの同時ユーザ (または同じプロジェクト内のユーザ) によって変更されている場合があります。このような場合、動的ではないデータベース検索により無効なデータを検索できます。現在の値を見つけるには DB Lookup Dynamic を使用します。

このコンポーネントの典型的なもう 1 つの使用例としては、ローカル・マシンで使用可能なメモリ容量を超える検索テーブルがあります。DB Lookup Dynamic コンポーネントを使用することにより、検索の速度は遅くなりますが、レコード単位で検索を実行するのでキャッシュ・メモリを必要としません。

DB Lookup Dynamic コンポーネントの設定

1　DB Lookup Dynamic コンポーネントを設計ウィンドウにドラッグします。

2　設計ウィンドウで、DB Lookup の IN ポートをインバウンド・データを提供するコンポーネントの OUT ポートに接続します。

3　プロパティ・ウィンドウでインタフェースとホスト名を指定します。

特定のフィールドの要件については、「DB Lookup Dynamic プロパティ・リスト」(155 ページ) を参照してください。

4　キー属性と値属性を指定します。これらが検索値です。

5　[Query] アイコンをクリックし、[Query] ウィンドウを開きます。

6　[Query] ウィンドウで検索データを取得するためのクエリを作成し、保存します。ソース・テーブルから検索値を返すようにクエリを設計します。

❖ 検索キー値の再設定

コンポーネントのキー属性と値属性のプロパティ値をデフォルトに再設定するには、次の手順に従います。

1　DB Lookup Dynamic コンポーネントを右クリックします。

2　[Reset Lookup Key] 値を選択します。

例

ドイツの製品に使用する製品番号を米国で使用する製品番号に置換するとします。ドイツの製品は、テーブル PRODUKTE(PR_NUMMER, PR_NAME, PR_PREIS) にあります。したがって、DB Lookup Dynamic コンポーネントの IN ポートにはそれらの 3 つの属性が含まれます。

米国の製品番号を検索するテーブルは、LOOKUP_PRODUCTS(SOURCE, DESTINATION) です。SOURCE カラムにはドイツの製品番号が、DESTINATION カラムには米国の製品番号が含まれます。

LOOKUP_PRODUCTS にドイツの PR_NUMMER の値が見つからない場合は、現在の PR_NUMMER は “INVALID” という文字列で置換されます。正常に実行された検索では、ドイツの製品番号が対応する米国の番号で置換されます。

この例の DB Lookup Dynamic コンポーネントを設定するには、以下を選択します。

- [Key Attribute]：PR_NUMMER
- [Value Attribute]：PR_NUMMER
- [Default Value]：INVALID
- [Query]：select DESTINATION FROM LOOKUP_PRODUCTS, where SOURCE = ‘[Lookup]’

DB Lookup Dynamic プロパティ・リスト

次の表には、DB Lookup Dynamic コンポーネントの必須プロパティおよびオプション・プロパティが示されています。必須プロパティは、**太字**テキストで表示されます。

必須プロパティ

プロパティ	説明
[Key Attribute]	IN ポート属性のリストからキー属性を選択します。この属性は、検索テーブルの最初のカラムに対応します。
[Value Attribute]	[Value Attribute] リストから、検索で見つけた値を割り当てる属性を選択します。返された検索値は、既存の値を上書きします。 キー属性と値属性の両方は、レコード構造の同じ属性を参照できます。それにより、対応する値でキーを上書きすることができます。
[Interface]	データ・ソースへの接続に使用するメソッドまたはドライバを指定します。
[Host Name]	データ・ソースを指定します。ホスト名リストにあるオプションは、選択したインタフェースによって異なります。 SQLite をホストとして使用する場合は、SQLite データベース・ファイルのパスおよびファイル名を入力します (たとえば、*c:¥mySQLite.db*)。指定した名前を持つデータベースが存在しない場合は、SQLite によって作成されます。
[Query]	[Query] アイコンをクリックして、データ・ソースから情報を取得するクエリを作成します。[Query] ウィンドウを使用して単純なクエリを作成するか、[Query Designer] アイコンをクリックして Query Designer を開きます。

オプション・プロパティ

プロパティ	説明
[User and Password]	認証されたデータベース・ユーザとパスワードを指定し、認証されていないアクセスからデータベースを保護します。
[Default Value]	キー値が検索テーブルに見つからない場合に、値属性に割り当てるデフォルト値を指定します。
[Use Key Value]	検索が失敗した場合に、デフォルトではなくキー値を値属性に割り当てるにはこのオプションを選択します。
[Lookup Empty/Null Keys]	空白または NULL のキー値の検索を実行する場合は、このオプションを選択します。このオプションを選択しない場合は、デフォルトの方法が適用されます。
[Lookup Key Value]	検索キーの値を指定します。
[Database]	データ・ソースとして使用するデータベースを指定します。さらに、適切なインタフェース、場合によっては適切なユーザ ID およびパスワードを選択する必要があります。
[Schema]	データ・ソースとして使用するスキーマ／所有者を識別します。表示されたオブジェクトは適切に制限され、新しいテーブルがそのスキーマに作成されます。
[Standardize Data Format]	異なるフォーマットをサポートするシステム間で Sybase ETL が移動できるように、受信する `DATE` および `NUMBER` 情報を標準フォーマットに変換するオプションを選択します。 日付は、年、月、日、時、分、秒、秒の小数点以下が含まれるフォーマット `YYYY-MM-DD hh:mm:ss.s` に変換されます。 例を示します。 `2005-12-01 16:40:59.123` 数値は、小数点として「.」を使用して変換されます。
[Database Options]	[Database Options] アイコンをクリックして、パフォーマンスのデフォルトを上書きし、一部のトランザクションの動作を制御するオプションを設定します。 詳細については、「データベース接続の設定」(105 ページ) を参照してください。

DB Lookup Dynamic のデモ

Sybase ETL には、DB Lookup Dynamic コンポーネントのデモが複数含まれています。これらのデモは、[Help] メニューの Flash デモ、および DemoRepository のサンプル・プロジェクトとして使用できます。

Flash デモ

Flash デモを実行するには、[Help] - [Demonstrations] - [Lookup] をクリックし、[DB Lookup - Dynamic] を選択します。

DemoRepository

ナビゲータで、[Repository] - [TRANSFORMER.transformer.Repository] - [Projects] の順に選択して、[Demo DB Lookup Dynamic] を選択します。

Staging コンポーネント

Staging コンポーネントには、IN ポートと OUT ポートが少なくとも 1 つずつあり、変換ストリームのデータに特定の変換が適用されます。

コンポーネント	説明
DB Staging	受信データ・ストリームを 1 つの作業領域にロードします。DB Staging は、すべての受信データをバッファし、送信データ・ストリームを作成します。これは、特定の select 文の結果セットを表します。

DB Staging

DB Staging は、受信データ・ストリームを 1 つの作業領域にロードする Staging コンポーネントです。DB Staging は、すべての受信データをバッファし、送信データ・ストリームを作成します。これは、特定の select 文の結果セットを表します。

前述のコンポーネントの出力ポート構造に基づいてステージング・テーブルを作成できます。多くの Transformation コンポーネンは、レコードごとに作業できるよう設計されていますが、ステージング・コンポーネントには次の 2 つのフェーズがあります。

- フェーズ 1：前述のコンポーネントからすべてのレコードを収集します。
- フェーズ 2：クエリを実行し、特定サイズのブロックに結果セットのレコードを提供します。

ステージング・コンポーネントを使用し、以前の順不同レコードで Query プロパティの ORDER BY 句または GROUP BY 句を使用して、ソートまたは集計できます。異種ソースからのデータは、ステージング・データベースの複数のテーブルにロードすることによりジョインさせることができます。このコンポーネントを使用して、検査または処理のために転送の中間イメージを作成できます。

シミュレーション・シーケンスへの影響

DB Staging コンポーネントは、最初に元のデータ・ソースからすべてのデータを取得し、後続のコンポーネントの新しいデータ・ソースとして動作することにより、シミュレーションの流れに影響を与えます。このコンポーネントは、元の Source コンポーネントの [Read Block Size] 値を上書きできます。

DB Staging コンポーネントの設定

1　DB Staging コンポーネントを設計ウィンドウにドラッグします。

2　DB Staging の IN ポートをインバウンド・データを提供するコンポーネントに接続します。

入力ストリームを DB Staging コンポーネントに追加するには、ステージング・コンポーネント上にコンポーネントを提供するデータからの接続を削除します。ポートはコンポーネントによって自動的に作成されます。

3　プロパティ・ウィンドウで、ステージング・テーブルを追加するデータベースに接続パラメータを追加します。

有効なインタフェースおよびホスト名を指定して、接続を作成します。特定のフィールドの要件については、「DB Staging プロパティ・リスト」(160 ページ) を参照してください。

注意 使用するステージング・テーブルがすでに存在する場合は、次の手順を省略してください。

4　設計ウィンドウで、DB Staging コンポーネントを右クリックし、次のオプションのいずれかを選択します。

- [Create Staging Table from Input]
- [Create Staging Table from Port]

5　[Create Staging Table from Input] オプションを選択した場合、適切なポート構造を選択し、[OK] をクリックします。新しいテーブルの名前を入力します。

[Create Staging Table from Port] オプションを選択した場合は、新しいテーブルの名前を入力して、適切なポート構造を選択します。[Apply] をクリックします。

6　[Add table] ウィンドウで、新しいテーブルを含む情報が適切かどうかを確認し、[Create] をクリックします。

7　プロパティ・ウィンドウで、[Stage Options] アイコンをクリックして [Stage Options] ウィンドウを開きます。

[Stage Options] ウィンドウでは、各ステージング・テーブルの [Truncate Table] および [Write Block Size] オプションも定義できます。

8　プロパティ・ウィンドウで、[Query] アイコンをクリックして、クエリを作成して作業領域からデータを選択します。

❖ **ポート構造の更新**

データベースが変更されたポート構造を更新するには、次の手順に従います。

1 DB Staging コンポーネントを右クリックします。

2 [Reconfigure] を選択します。

[Reconfigure] オプションを選択すると、データベース・スキーマが変更されている場合、コンポーネントの設定が更新されます。現在の接続が閉じ、データベースへの新しい接続が開かれ、クエリのメタデータが読み取られ、ポート構造の更新が適用されます。

DB Staging プロパティ・リスト

次の表には、Data Staging コンポーネントの必須プロパティおよびオプション・プロパティが示されています。必須プロパティは、**太字**テキストで表示されます。

必須プロパティ

プロパティ	説明
[Interface]	データ・ソースへの接続に使用するメソッドまたはドライバを指定します。
[Host Name]	データ・ソースを指定します。ホスト名リストにあるオプションは、選択したインタフェースによって異なります。 SQLite をホストとして使用する場合は、SQLite データベース・ファイルのパスおよびファイル名を入力します (たとえば、*c:¥mySQLite.db*)。指定した名前を持つデータベースが存在しない場合は、SQLite によって作成されます。
[Stage Options]	[Stage Options] アイコンをクリックして、ステージング・オプションを設定します。
[Query]	[Query] アイコンをクリックして、データ・ソースから情報を取得するクエリを作成します。[Query] ウィンドウを使用して単純なクエリを作成するか、[Query Designer] アイコンをクリックしてクエリを生成します。

オプション・プロパティ

プロパティ	説明
[User and Password]	認証されたデータベース・ユーザとパスワードを指定し、認証されていないアクセスからデータベースを保護します。
[Read Block Size]	1 つのステップでコンポーネントが取得するレコードの数を決定します。
[Pre-processing SQL]	コンポーネントの初期化時に実行するクエリを作成する場合、[Pre-processing SQL] アイコンをクリックします。 クエリには 1 つまたは複数の SQL 文を含めることができます。複数の文を使用する場合は、セミコロン (;) で区切ります。
[Post-Processing SQL]	すべてのコンポーネントの実行後に実行するクエリを作成する場合、[Post-processing SQL] アイコンをクリックします。 クエリには 1 つまたは複数の SQL 文を含めることができます。複数の文を使用する場合は、セミコロン (;) で区切ります。
[Database]	データ・ソースとして使用するデータベースを指定します。さらに、適切なインタフェース、場合によっては適切なユーザ ID およびパスワードを選択する必要があります。
[Schema]	データ・ソースとして使用するスキーマ／所有者を識別します。表示されたオブジェクトは適切に制限され、新しいテーブルがそのスキーマに作成されます。
[Standardize Data Format]	受信する DATE および NUMBER 情報を、異なるフォーマットをサポートするシステム間で Sybase ETL が移動できる標準フォーマットに変換するオプションを選択します。 日付は、年、月、日、時、分、秒、秒の小数点以下が含まれるフォーマット YYYY-MM-DD hh:mm:ss.s に変換されます。 例を示します。 `2005-12-01 16:40:59.123` 数値は、小数点として「.」を使用して変換されます。

プロパティ	説明
[IQ Lock Table in Exclusive Mode]	ターゲット・テーブルをロックし、同時トランザクションによる更新を防ぐオプションを選択します。排他テーブル・ロックが適用されると、他のトランザクションはロック・テーブルに対してクエリの発行またはあらゆる修正を実行できません。 また、[IQ Lock Table in Exclusive Mode] オプションによって、Sybase IQ 内の同じテーブルをロードする複数のプロジェクトがキューに入れられます。
[Wait Time for IQ Lock Table]	Exclusive ロックを取得する前に、プロジェクトが待機する最長ブロック時間を指定します。 time 引数は、hh:nn:ss.sss の形式で指定します。time 引数を入力しないと、Exclusive ロックが使用できるようになるか、または割り込みが発生するまで、サーバはいつまでも待機します。「00:00:00.000」を time 引数に指定すると、プロジェクトの開始時に Exclusive ロックが取得されます。
[Database Options]	[Database Options] アイコンをクリックして、パフォーマンスのデフォルトを上書きし、一部のトランザクションの動作を制御するオプションを設定します。これらのプロパティのリストを参照してください。 詳細については、「データベース接続の設定」(105 ページ) を参照してください。

DB Staging デモ

Sybase ETL には、DB Staging コンポーネントのデモが複数含まれています。これらのデモは、[Help] メニューの Flash デモ、および DemoRepository のサンプル・プロジェクトとして使用できます。

Flash デモ

Flash デモを実行するには、[Help] - [Demonstrations] - [Staging] の順にクリックし、[DB Staging] を選択します。

DemoRepository

ナビゲータで、[Repository] - [TRANSFORMER.transformer.Repository] - [Projects] の順に選択して、[Demo DB Staging] を選択します。

Destination コンポーネント

Destination コンポーネント (データ・シンクとも呼ばれる) は、特定のターゲットにデータを書き込みます。このコンポーネント・タイプには、1 つの IN ポートがありますが、OUT ポートはありません。

コンポーネント	説明
DB Data Sink Insert	データベース・テーブルにレコードを追加します。属性を除外、またはデフォルト値を割り当て、テーブルに挿入するレコードを決定します。 このコンポーネントを使用して、コンポーネントの IN ポートからのすべてのレコードをデータベース・テーブルに追加します。
DB Data Sink Update	選択したキーに一致するすべてのレコードを更新または上書きします。このコンポーネントは、新しいレコードを挿入できません。 このコンポーネントを使用して既存のレコードを更新します。新しいレコードは挿入しません。
DB Data Sink Delete	DB Data Sink Delete は、選択したキーの受信値に一致する送信先テーブルからのレコードを削除します。 このコンポーネントを使用して、送信先テーブルからレコードを削除します。
Text Data Sink	Text Data Sink は、区切りフォーマットまたは固定長フォーマットで、変換結果をテキスト・ファイルに書き込みます。
DB Bulk Load Sybase IQ	DB Bulk Load Sybase IQ は、Sybase IQ テーブルでバルク操作を実行します。 このコンポーネントを使用して、コンポーネントの IN ポートからのレコードに基づいて、Sybase IQ データベースのテーブルのレコードを操作します。

DB Data Sink Insert

DB Data Sink Insert は、IN ポートからのレコードをデータベース・テーブルに追加する Destination コンポーネントです。属性を除外、またはデフォルト値を割り当て、テーブルに挿入するレコードを決定します。

DB Data Sink Insert コンポーネントの設定

1 DB Data Sink Insert コンポーネントを設計ウィンドウにドラッグします。

2 [Database Configuration] ウィンドウで、ターゲット・データベースの接続パラメータを追加します。

3 特定のフィールドの要件については、「DB Data Sink Insert プロパティ・リスト」(166 ページ) を参照してください。

4 ロード先テーブルを選択または入力します。

既存のテーブルに書き込むか、プロジェクトの既存のポートに基づいてテーブルを追加できます。既存のポート構造に基づいてテーブルを追加する場合は、この手順を省略してください。詳細については、「送信先テーブルの追加」(165 ページ) を参照してください。

5 [Finish] をクリックします。

6 プロパティ・ウィンドウで他のオプション・プロパティを指定します。

❖ **ポート構造の更新**

データベースが変更されたポート構造を更新するには、次の手順に従います。

1 DB Data Sink Insert コンポーネントを右クリックします。

2 [Reconfigure] を選択します。

[Reconfigure] オプションを選択すると、データベース・スキーマが変更されている場合、コンポーネントの設定が更新されます。現在の接続が閉じ、データベースへの新しい接続が開かれ、クエリのメタデータが読み取られ、ポート構造の更新が適用されます。

❖ **入力ポートでのデータのロード**

入力ポートで現在使用可能なデータをデータベースまたはテキスト・ファイルにロードするには、次の手順に従います。

1　DB Data Sink Insert コンポーネントを右クリックします。

2　[Flush Buffer] を選択します。

[Flush Buffer] オプションを選択すると、バッファされたローがターゲットに書き込まれます。書き込みブロック・サイズが指定されているすべてのコンポーネントは、書き込みブロック・サイズに達するまでデータをバッファします。シミュレーション中は、バッファ中のデータが存在する場合、[Flush Buffer] オプションはターゲットに書き込まれていないローの数と共に表示されます。バッファが空の状態になると、このオプションは無効になります。

送信先テーブルの追加

変換結果を既存のテーブルに書き込むか、プロジェクトの既存のポートに基づいて送信先テーブルを追加します。[Database Configuration] ウィンドウまたはプロパティ・ウィンドウからのポート構造に基づいてテーブルを追加することはできません。設計ウィンドウでポート構造を選択する必要があります。

❖ **既存のテーブルへの書き込み**

1　[Database Configuration] ウィンドウで、変換結果を書き込むテーブルを指定します。[Destination Table] アイコンをクリックしてテーブルを選択するか、[Destination Table] フィールドにテーブル名を手動で入力します。

2　[Finish] をクリックします。

❖ **IN ポートに基づいた送信先テーブルの追加**

1　設計ウィンドウで、コンポーネントを右クリックし、[Add Destination Table from Input] を選択します。

2　新しいテーブルに名前を付けます。[OK] をクリックします。

3　[Add table] ウィンドウでテーブル情報が正しいことを確認し、[Create] をクリックします。

❖ **既存のポートからの送信先テーブルの追加**

1 設計ウィンドウで、コンポーネントを右クリックし、[Add Destination Table from port] を選択します。

2 新しいテーブルに名前を付けます。[OK] をクリックします。

3 新しいテーブルに割り当てる構造を持つポートを選択します。[Apply] をクリックします。

4 [Add table] ウィンドウでテーブル情報が正しいことを確認し、[Create] をクリックします。

注意 ツールセットを使用して送信先テーブルを作成するか、[Properties] ウィンドウから既存のテーブルを選択します。

DB Data Sink Insert プロパティ・リスト

次の表には、DB Data Sink Insert コンポーネントの必須プロパティおよびオプション・プロパティが示されています。必須プロパティは、**太字**テキストで表示されます。

必須プロパティ

プロパティ	説明
[Interface]	データ・ソースへの接続に使用するメソッドまたはドライバを指定します。
[Host Name]	データ・ソースを指定します。ホスト名リストにあるオプションは、選択したインタフェースによって異なります。 SQLite をホストとして使用する場合は、SQLite データベース・ファイルのパスおよびファイル名を入力します (たとえば、*c:¥mySQLite.db*)。指定した名前を持つデータベースが存在しない場合は、SQLite によって作成されます。
[Destination Table]	[Destination Table] アイコンをクリックして既存のテーブルのセットから送信先テーブルを選択するか、送信先テーブルを手動で入力します。 コンポーネントのポート構造に基づいて、新しい送信先テーブルも作成できます。詳細については、「送信先テーブルの追加」(165 ページ) を参照してください。

オプション・プロパティ

プロパティ	説明
[User and Password]	認証されたデータベース・ユーザとパスワードを指定し、認証されていないアクセスからデータベースを保護します。
[Insert Options]	[Insert Options] アイコンをクリックして、属性の挿入方法を決定します。Include カラムは、挿入する属性を指定します。SQL 式は、実行時の式の値で属性値を上書きします。 選択した属性の受信値を上書きする各属性に対し INSERT 文を指定します。SQL INSERT 値句の例は次のとおりです。 `'valid' '[uDate("now")]'` レコードは、[Insert Options] ウィンドウで選択されたすべての属性とともに挿入されます。属性を選択、または選択を解除します。挿入される属性の値を指定できます。つまり受信属性の対応する値を上書きします。
[Truncate Table]	変換プロセスの初期化の際に、送信先テーブルからすべてのレコードを削除するオプションを選択します。
[Write Block Size]	1 回の書き込み操作でファイルまたはパイプに書き込まれるレコードの数を指定します。
[Pre-processing SQL]	コンポーネントの初期化時に実行するクエリを作成する場合、[Pre-processing SQL] アイコンをクリックします。 クエリには 1 つまたは複数の SQL 文を含めることができます。複数の文を使用する場合は、セミコロン (;) で区切ります。
[Post-Processing SQL]	すべてのコンポーネントの実行後に実行するクエリを作成する場合、[Post-processing SQL] アイコンをクリックします。 クエリには 1 つまたは複数の SQL 文を含めることができます。複数の文を使用する場合は、セミコロン (;) で区切ります。
[Opening Attribute Quote]	属性名のプレフィクスを SQL 文で指定します。
[Closing Attribute Quote]	属性名のポストフィクスを SQL 文で指定します。
[Database]	データ・ソースとして使用するデータベースを選択します。 さらに、適切なインタフェース、場合によっては適切なユーザ ID およびパスワードを選択する必要があります。

プロパティ	説明
[Schema]	データ・ソースとして使用するスキーマ／所有者を識別します。表示されたオブジェクトは適切に制限され、新しいテーブルがそのスキーマに作成されます。
[Standardize Data Format]	異なるフォーマットをサポートするシステム間で Sybase ETL が移動できるように、受信する DATE および NUMBER 情報を標準フォーマットに変換するオプションを選択します。 日付は、年、月、日、時、分、秒、秒の小数点以下が含まれるフォーマット YYYY-MM-DD hh:mm:ss.s に変換されます。 例を示します。 2005-12-01 16:40:59.123 数値は、小数点として「.」を使用して変換されます。
[IQ Lock Table in Exclusive Mode]	ターゲット・テーブルをロックし、同時トランザクションによる更新を防ぐオプションを選択します。排他テーブル・ロックが適用されると、他のトランザクションはロック・テーブルに対してクエリの発行またはあらゆる修正を実行できません。 また、[IQ Lock Table in Exclusive Mode] オプションによって、Sybase IQ 内の同じテーブルをロードする複数のプロジェクトがキューに入れられます。
[Wait Time for IQ Lock Table]	Exclusive ロックを取得する前に、プロジェクトが待機する最長ブロック時間を指定します。 time 引数は、hh:nn:ss.sss の形式で指定します。time 引数を入力しないと、Exclusive ロックが使用できるようになるか、または割り込みが発生するまで、サーバはいつまでも待機します。「00:00:00.000」を time 引数に指定すると、プロジェクトの開始時に Exclusive ロックが取得されます。
[Database Options]	[Database Options] アイコンをクリックして、パフォーマンスのデフォルトを上書きし、一部のトランザクションの動作を制御するオプションを設定します。 詳細については、「データベース接続の設定」(105 ページ) を参照してください。

DB Data Sink Insert デモ

Sybase ETL には、DB Data Sink Insert コンポーネントのデモが複数含まれています。これらのデモは、[Help] メニューの Flash デモ、および DemoRepository のサンプル・プロジェクトとして使用できます。

Flash デモ

Flash デモを実行するには、[Help] - [Demonstrations] - [Destination] の順にクリックし、[DB Data Sink - Insert] を選択します。

DemoRepository

ナビゲータで、[Repository] - [TRANSFORMER.transformer.Repository] - [Projects] をクリックして、次を選択します。

- [Demo Transfer German Customers]
- [Demo Transfer German Products]
- [Demo Transfer German Sales]
- [Demo Transfer U.S. Customers]
- [Demo Transfer U.S. Products]

DB Data Sink Update

DB Data Sink Update は、選択したキーと一致するすべてのレコードを更新または上書きする Destination コンポーネントです。このコンポーネントは、新しいレコードを挿入できません。一致するレコードがない場合、DB Data Sink Update は、エラー・メッセージを表示しません。

このコンポーネントを使用して既存のレコードを更新します。新しいレコードは挿入しません。

注意　更新値が、制約、参照整合性、またはユニーク・インデックス定義など基本となるテーブルまたはオブジェクトの制限を超えると、エラー・メッセージが表示されます。選択されたキー値属性は、既存のインデックス定義に依存しません。

DB Data Sink Update コンポーネントの設定

1 DB Data Sink Update を設計ウィンドウにドラッグします。

2 [Database Configuration] ウィンドウで、ターゲット・データベースの接続パラメータを追加します。

特定のフィールドの要件については、「DB Data Sink Update プロパティ・リスト」(171 ページ) を参照してください。

3 変換結果を書き込む送信先テーブルを指定します。[Destination Table] アイコンをクリックしてテーブルを選択するか、[Destination Table] フィールドにテーブル名を手動で入力します。

既存のテーブルに書き込むか、プロジェクトの既存のポートに基づいてテーブルを追加できます。既存のポート構造に基づいてテーブルを追加する場合は、この手順を省略してください。詳細については、「送信先テーブルの追加」(171 ページ) を参照してください。

4 [Finish] をクリックします。

5 プロパティ・ウィンドウで、[Key] アイコンをクリックし、更新するレコードを識別する送信先テーブルのカラムを選択します。

キーを選択する前に送信先テーブルを指定する必要があります。その後で複数のキー・カラムを選択できます。これは論理的な選択です。データベース・スキーマの基本となるインデックスとは関係ありません。

6 プロパティ・ウィンドウで他のオプション・プロパティを指定します。

❖ ポート構造の更新

データベースが変更されたポート構造を更新するには、次の手順に従います。

1 DB Data Sink Update コンポーネントを右クリックします。

2 [Reconfigure] を選択します。

[Reconfigure] オプションを選択すると、データベース・スキーマが変更されている場合、コンポーネントの設定が更新されます。現在の接続が閉じ、データベースへの新しい接続が開かれ、クエリのメタデータが読み取られ、ポート構造の更新が適用されます。

❖ **入力ポートでのデータのロード**

入力ポートで現在使用可能なデータをデータベースまたはテキスト・ファイルにロードするには、次の手順に従います。

1　DB Data Sink Update コンポーネントを右クリックします。

2　[Flush Buffer] を選択します。

[Flush Buffer] オプションを選択すると、バッファされたローがターゲットに書き込まれます。書き込みブロック・サイズが指定されているすべてのコンポーネントは、書き込みブロック・サイズに達するまでデータをバッファします。シミュレーション中は、バッファ中のデータが存在する場合、[Flush Buffer] オプションはターゲットに書き込まれていないローの数と共に表示されます。バッファが空の状態になると、このオプションは無効になります。

送信先テーブルの追加

変換結果を既存のテーブルに書き込むか、プロジェクトの既存のポートに基づいて送信先テーブルを追加します。送信先テーブルの追加方法については、「送信先テーブルの追加」(165 ページ) を参照してください。

DB Data Sink Update プロパティ・リスト

次の表には、DB Data Sink Update コンポーネントの必須プロパティおよびオプション・プロパティが示されています。必須プロパティは、**太字**テキストで表示されます。

必須プロパティ

プロパティ	説明
[Interface]	データ・ソースへの接続に使用するメソッドまたはドライバを指定します。
[Host Name]	データ・ソースを指定します。ホスト名リストにあるオプションは、選択したインタフェースによって異なります。 SQLite をホストとして使用する場合は、SQLite データベース・ファイルのパスおよびファイル名を入力します (たとえば、*c:¥mySQLite.db*)。指定した名前を持つデータベースが存在しない場合は、SQLite によって作成されます。

プロパティ	説明
[Destination Table]	既存のテーブルのセットからロード先テーブルを選択する場合、[Destination Table] アイコンをクリックします。 コンポーネントのポート構造に基づいて、新しい送信先テーブルも作成できます。詳細については、「送信先テーブルの追加」(171 ページ) を参照してください。
[Key]	更新するレコードを識別する送信先テーブルのカラムを選択します。 キーを選択する前に送信先テーブルを選択する必要があります。その後で複数のキー・カラムを選択できます。

オプション・プロパティ

プロパティ	説明
[User and Password]	認証されたデータベース・ユーザとパスワードを指定し、認証されていないアクセスからデータベースを保護します。

[Update Options]	[Update Options] アイコンをクリックし、更新に含める属性 (キー属性はリストされません) を選択します。すべての属性はデフォルトで選択されています。更新から除外する属性の選択を解除します。
	SQL UPDATE SET 句カラムでは、受信属性の値を新しい値で上書きできます。新しい値には、定数または式を指定できます。
	SQL 言語表記では、カラムのコンテンツは次のように処理されます。
	`UPDATE customers`
	`SET cu_createdate = '2005-02-26'`
	`WHERE ….`
	基本となるデータベースの SQL 言語で許可されるいずれの式も使用できます。角カッコ内の動的式は、コンポーネントの初期化中に評価されます。

[Write Block Size]	1 回の書き込み操作でファイルまたはパイプに書き込まれるレコードの数を指定します。

プロパティ	説明
[Pre-processing SQL]	[Pre-processing SQL] アイコンをクリックして、コンポーネントの初期化中に実行されるクエリを作成できるウィンドウを開きます。 クエリには 1 つまたは複数の SQL 文を含めることができます。複数の文を使用する場合は、セミコロン (;) で区切ります。
[Post-Processing SQL]	[Post-processing SQL] アイコンをクリックし、すべてのコンポーネントの実行後に実行されるクエリを作成できるウィンドウを開きます。 クエリには 1 つまたは複数の SQL 文を含めることができます。複数の文を使用する場合は、セミコロン (;) で区切ります。
[Opening Attribute Quote]	属性名のプレフィクスを SQL 文で指定します。
[Closing Attribute Quote]	属性名のポストフィクスを SQL 文で指定します。
[Database]	データ・ソースとして使用するデータベースを指定します。 さらに、適切なインタフェース、場合によっては適切なユーザ ID およびパスワードを選択する必要があります。
[Schema]	データ・ソースとして使用するスキーマ／所有者を識別します。表示されたオブジェクトは適切に制限され、新しいテーブルがそのスキーマに作成されます。
[Standardize Data Format]	異なるフォーマットをサポートするシステム間で Sybase ETL が移動できるように、受信する `DATE` および `NUMBER` 情報を標準フォーマットに変換するオプションを選択します。 日付は、年、月、日、時、分、秒、秒の小数点以下が含まれるフォーマット `YYYY-MM-DD hh:mm:ss.s` に変換されます。 例を示します。 `2005-12-01 16:40:59.123` 数値は、小数点として「.」を使用して変換されます。

プロパティ	説明
[IQ Lock Table in Exclusive Mode]	ターゲット・テーブルをロックし、同時トランザクションによる更新を防ぐオプションを選択します。排他テーブル・ロックが適用されると、他のトランザクションはロック・テーブルに対してクエリの発行またはあらゆる修正を実行できません。 また、[IQ Lock Table in Exclusive Mode] オプションによって、Sybase IQ 内の同じテーブルをロードする複数のプロジェクトがキューに入れられます。
[Wait Time for IQ Lock Table]	Exclusive ロックを取得する前に、プロジェクトが待機する最長ブロック時間を指定します。 time 引数は、hh:nn:ss.sss の形式で指定します。time 引数を入力しないと、Exclusive ロックが使用できるようになるか、または割り込みが発生するまで、サーバはいつまでも待機します。「00:00:00.000」を time 引数に指定すると、プロジェクトの開始時に Exclusive ロックが取得されます。
[Database Options]	[Database Options] アイコンをクリックして、パフォーマンスのデフォルトを上書きし、一部のトランザクションの動作を制御するオプションを設定します。 詳細については、「データベース接続の設定」(105 ページ) を参照してください。

DB Data Sink Update デモ

Sybase ETL には、DB Data Sink Update コンポーネントのデモが複数含まれています。これらのデモは、[Help] メニューの Flash デモ、および DemoRepository のサンプル・プロジェクトとして使用できます。

Flash デモ

Flash デモを実行するには、[Help] - [Demonstrations] - [Destination] の順にクリックし、[DB Data Sink - Update] を選択します。

DemoRepository

ナビゲータで、[Repository] - [TRANSFORMER.transformer.Repository] - [Projects] の順に選択して、[Demo DB Datasink Update] を選択します。

DB Data Sink Delete

DB Data Sink Delete は、選択したキーの受信値と一致するデータベース送信先テーブルからレコードを削除する Destination コンポーネントです。一致するレコードがない場合、DB Data Sink Delete はエラー・メッセージを表示しません。

DB Data Sink Delete コンポーネントの設定

1　DB Data Sink Delete コンポーネントを設計ウィンドウにドラッグします。

2　[Database Configuration] ウィンドウで、ターゲット・データベースの接続パラメータを追加します。有効なインタフェースおよびホスト名を指定します。

特定のフィールドの要件については、「DB Data Sink Delete プロパティ・リスト」(176 ページ) を参照してください。

3　変換結果を書き込むテーブルを指定します。[Destination Table] アイコンをクリックしてテーブルを選択するか、[Destination Table] フィールドにテーブル名を手動で入力します。

既存のテーブルに書き込むか、プロジェクトの既存のポートに基づいてテーブルを追加できます。既存のポート構造に基づいてテーブルを追加する場合は、この手順を省略してください。詳細については、「送信先テーブルの追加」(176 ページ) を参照してください。

4　[Finish] をクリックします。

5　プロパティ・ウィンドウで、[Key] アイコンをクリックし、送信先テーブルから削除するレコードを識別するカラムを選択します。

キーを選択する前に送信先テーブルを選択する必要があります。その後で複数のキー・カラムを選択できます。これは論理的な選択です。データベース・スキーマの基本となるインデックスとは関係ありません。

6　プロパティ・ウィンドウで他のオプション・プロパティを指定します。

❖ **ポート構造の更新**

データベースが変更されたポート構造を更新するには、次の手順に従います。

1 DB Data Sink Delete コンポーネントを右クリックします。

2 [Reconfigure] を選択します。

[Reconfigure] オプションを選択すると、データベース・スキーマが変更されている場合、コンポーネントの設定が更新されます。現在の接続が閉じ、データベースへの新しい接続が開かれ、クエリのメタデータが読み取られ、ポート構造の更新が適用されます。

❖ **入力ポートでのデータのロード**

入力ポートで現在使用可能なデータをデータベースまたはテキスト・ファイルにロードするには、次の手順に従います。

1 DB Data Sink Delete コンポーネントを右クリックします。

2 [Flush Buffer] を選択します。

[Flush Buffer] オプションを選択すると、バッファされたローがターゲットに書き込まれます。書き込みブロック・サイズが指定されているすべてのコンポーネントは、書き込みブロック・サイズに達するまでデータをバッファします。シミュレーション中は、バッファ中のデータが存在する場合、[Flush Buffer] オプションはターゲットに書き込まれていないローの数と共に表示されます。バッファが空の状態になると、このオプションは無効になります。

送信先テーブルの追加

変換結果を既存のテーブルに書き込むか、プロジェクトの既存のポートに基づいて送信先テーブルを追加します。送信先テーブルの追加方法については、「送信先テーブルの追加」(165 ページ) を参照してください。

DB Data Sink Delete プロパティ・リスト

次の表には、DB Data Sink Delete コンポーネントの必須プロパティおよびオプション・プロパティが示されています。必須プロパティは、**太字**テキストで表示されます。

必須プロパティ

プロパティ	説明
[Interface]	データ・ソースへの接続に使用するメソッドまたはドライバを識別します。
[Host Name]	データ・ソースを識別します。ホスト名リストにあるオプションは、選択したインタフェースによって異なります。 SQLite をホストとして使用する場合は、SQLite データベース・ファイルのパスおよびファイル名を入力します (たとえば、*c:¥mySQLite.db*)。指定した名前を持つデータベースが存在しない場合は、SQLite によって作成されます。
[Destination Table]	[Destination Table] アイコンをクリックして、既存のテーブルのセットから送信先テーブルを選択できるウィンドウを開きます。 コンポーネントのポート構造に基づいて、新しい送信先テーブルも作成できます。詳細については、「送信先テーブルの追加」(176 ページ) を参照してください。
[Key]	削除するレコードを識別する送信先テーブルのカラムを選択します。 キーを選択する前に送信先テーブルを選択する必要があります。その後で複数のキー・カラムを選択できます。

オプション・プロパティ

プロパティ	説明
[User and Password]	認証されたデータベース・ユーザを識別する組み合わせに使用され、認証されていないアクセスからデータベースを保護します。
[Write Block Size]	1 回の書き込み操作でファイルに書き込まれるレコードの数を指定します。
[Pre-processing SQL]	[Pre-processing SQL] アイコンをクリックして、コンポーネントの初期化中に実行されるクエリを作成できるウィンドウを開きます。 クエリには 1 つまたは複数の SQL 文を含めることができます。複数の文を使用する場合は、セミコロン (;) で区切ります。

プロパティ	説明
[Post-Processing SQL]	[Post-processing SQL] アイコンをクリックし、すべてのコンポーネントの実行後に実行されるクエリを作成できるウィンドウを開きます。 クエリには 1 つまたは複数の SQL 文を含めることができます。複数の文を使用する場合は、セミコロン (;) で区切ります。
[Opening Attribute Quote]	SQL 文での属性名のプレフィクスです。
[Closing Attribute Quote]	SQL 文での属性名のポストフィクスです。
[Database]	データ・ソースとして使用するデータベースを識別します。 このオプションを選択すると、適切なインタフェース、場合によっては適切なユーザ ID およびパスワードを選択する必要があります。
[Schema]	データ・ソースとして使用するスキーマ／所有者を識別します。表示されたオブジェクトは適切に制限され、新しいテーブルがそのスキーマに作成されます。
[Standardize Data Format]	異なるフォーマットをサポートするシステム間で Sybase ETL が移動できるように、受信する `DATE` および `NUMBER` 情報を標準フォーマットに変換します。 日付は、年、月、日、時、分、秒、秒の小数点以下が含まれるフォーマット `YYYY-MM-DD hh:mm:ss.s` に変換されます。 例を示します。 `2005-12-01 16:40:59.123` 数値は、小数点として「.」を使用して変換されます。
[IQ Lock Table in Exclusive Mode]	ターゲット・テーブルをロックし、同時トランザクションによる更新を防ぐオプションを選択します。排他テーブル・ロックが適用されると、他のトランザクションはロック・テーブルに対してクエリの発行またはあらゆる修正を実行できません。 また、[IQ Lock Table in Exclusive Mode] オプションによって、Sybase IQ 内の同じテーブルをロードする複数のプロジェクトがキューに入れられます。

プロパティ	説明
[Wait Time for IQ Lock Table]	Exclusive ロックを取得する前に、プロジェクトが待機する最長ブロック時間を指定します。 time 引数は、hh:nn:ss.sss の形式で指定します。time 引数を入力しないと、Exclusive ロックが使用できるようになるか、または割り込みが発生するまで、サーバはいつまでも待機します。「00:00:00.000」を time 引数に指定すると、プロジェクトの開始時に Exclusive ロックが取得されます。
[Database Options]	[Database Options] アイコンをクリックすると、パフォーマンスのデフォルトを上書きし、一部のトランザクションの動作を制御するオプションを設定できるウィンドウが開きます。 詳細については、「データベース接続の設定」(105 ページ) を参照してください。

DB Data Sink Delete デモ

Sybase ETL には、DB Data Sink Delete コンポーネントのデモが複数含まれています。これらのデモは、[Help] メニューの Flash デモ、および DemoRepository のサンプル・プロジェクトとして使用できます。

Flash デモ

Flash デモを実行するには、[Help] - [Demonstrations] - [Destination] の順にクリックし、[DB Data Sink - Delete] を選択します。

DemoRepository

ナビゲータで、[Repository] - [TRANSFORMER.transformer.Repository] - [Projects] の順に選択して、[Demo DB Datasink Delete] を選択します。

Text Data Sink

Text Data Sink は、区切られたフォーマットまたは固定長フォーマットでテキスト・ファイルに変換結果を書き込む Destination コンポーネントです。

Text Data Sink コンポーネントの設定

1 Text Data Sink コンポーネントを設計ウィンドウにドラッグします。

2 Text Data Sink の OUT ポートは、プロジェクトに追加する際にインバウンド・データを提供するコンポーネントの IN ポートにリンクする必要があります。

特定のフィールドの要件については、「Text Data Sink プロパティリスト」(185 ページ) を参照してください。また、次も参照してください。

- ファイル定義のエクスポートおよびインポート – コンポーネント・ウィンドウのエクスポートおよびインポート・オプションを使用すると、ファイル・プロパティを定義ファイルに保存し、他のコンポーネントから再使用できます。
- ポート構造 (デリミタファイル) の変更 – デリミタファイルのカラム値は、現在の IN ポート構造を反映します。別のポートに基づいてポート構造を割り当てるか、現在のポート構造を再作成できます。
- 固定長ファイルの操作 – 固定長ファイル・タイプを操作する場合、カラムを作成し、各カラムに対し位置パラメータを提供する必要があります。

3 [Save] をクリックします。

4　プロパティ・ウィンドウで他のオプション・プロパティを指定します。

注意 設計ウィンドウで使用できる隣接するコンポーネントがある場合、Sybase ETL は自動的に Text Data Sink の IN ポートとそのコンポーネントの OUT ポート間にリンクを作成します。このリンクは、デリミタファイルに対し、ウィンドウが開く際の初期ポート構造を提供します。

そうでない場合、設計ウィンドウを閉じて、Text Data Sink IN ポートと隣接するコンポーネントの OUT ポートを接続する必要があります。

注意 Text Data Sink は、シミュレーション・シーケンスに影響を与えません。

❖ 入力ポートでのデータのロード

入力ポートで現在使用可能なデータをデータベースまたはテキスト・ファイルにロードするには、次の手順に従います。

1　Text Data Sink コンポーネントを右クリックします。

2　[Flush Buffer] を選択します。

[Flush Buffer] オプションを選択すると、バッファされたローがターゲットに書き込まれます。書き込みブロック・サイズが指定されているすべてのコンポーネントは、書き込みブロック・サイズに達するまでデータをバッファします。シミュレーション中は、バッファ中のデータが存在する場合、[Flush Buffer] オプションはローの数と共に表示されます。バッファが空の状態になると、このオプションは無効になります。

ファイル定義のエクスポートおよびインポート

コンポーネント・ウィンドウのエクスポートおよびインポート・オプションを使用すると、ファイル・プロパティを定義ファイルに保存し、他のコンポーネントから再使用できます。エクスポートは、コンポーネント・プロパティを定義ファイルに保存します。インポートは、Export コマンドで作成した定義ファイルをロードします。

❖ ファイル定義のエクスポート

1 [Text Data Sink] をダブルクリックして、コンポーネント・ウィンドウを開きます。

2 [Properties] - [Export] の順にクリックします。

3 使用する定義ファイルを選択します。

❖ ファイル定義のインポート

1 [Text Data Sink] をダブルクリックして、コンポーネント・ウィンドウを開きます。

2 [Properties] - [Import] の順にクリックします。

3 使用する定義ファイルを選択します。

ポート構造 (デリミタファイル) の変更

[Text Data Sink Components] ウィンドウのカラム値は、現在の IN ポートの構造を反映します。新しいポート構造を割り当てるか、現在のポート構造を再作成できます。

❖ 新しいポート構造の割り当て

1　[Text Data Sink] をダブルクリックして、コンポーネント・ウィンドウを開きます。

2　[Column Names] パネルの [Assign Port Structure] アイコンをクリックします。

3　割り当てる構造を持つポートを選択します。

❖ カラム定義の再生成

1　[Text Data Sink] をダブルクリックして、コンポーネント・ウィンドウを開きます。

2　[Column Names] パネルの [Regenerate the column definition] アイコンを右クリックします。

固定長ファイルの操作

固定長ファイル・タイプの場合、カラムを作成し、各カラムに対し位置パラメータを提供する必要があります。

❖ 出力へのカラムの追加

1 [Text Data Sink] をダブルクリックして、コンポーネント・ウィンドウを開きます。

2 [Column Names] パネルの [Insert a New Attribute] アイコンをクリックします。生成されたカラムの名前を編集できます。

❖ 出力からカラムを削除するには

1 [Text Data Sink] をダブルクリックして、コンポーネント・ウィンドウを開きます。

2 カラムを選択して、[Remove an attribute] アイコンをクリックします。

Text Data Sink プロパティ リスト

次の表には、Text Data Sink コンポーネントの必須プロパティおよびオプション・プロパティが示されています。必須プロパティ・ラベルは、**太字**テキストで表示されます。

必須プロパティ

プロパティ	説明
[Text Destination]	出力ファイルを指定します。 Text Data Sink は、コンポーネントをプロジェクトに追加すると、送信先ファイルの入力を要求します。送信先ファイルを指定するには、プロパティ・ウィンドウの [Destination File] アイコンをクリックし、既存のファイルを選択するか、プロジェクトの実行中にフル・パスおよびファイル名を入力し、ファイルを作成します。
[Columns]	[Columns] アイコンをクリックして、ソース・ファイルのデータにカラムを定義します。プロパティ値が空でない場合、[Columns] 値は コンポーネント・ウィンドウで定義したポート構造または属性値を反映します。

オプション・プロパティ

プロパティ	説明
[Row Delimiter]	各ローの区切り方を選択します。 • [Position] (固定行位置) • [LF] (改行) • [CR] (行頭復帰) • [CRLF] (行頭復帰とそれに続く改行) 別のデリミタ文字を使用することもできます。
[Row Length]	ロー・デリミタとして [Position] を選択した場合、各固定ローの文字数を指定します。
[Column Delimiter]	カラムの区切り方を選択します。 • [Position] (固定カラム位置) • [Tab] • [Comma] • [Semicolon] 別のデリミタ文字を使用することもできます。

プロパティ	説明
[Column Quote]	出力ファイルの値に引用符を付ける方法を選択します (デリミタファイルのみ)。 • [None] • [Single quote] • [Double quote]
[Fixed by Bytes]	行の長さ、カラムの始まりとカラムの終わりに指定された値を解釈する方法を指定します。 • オフ – 値は、文字数として解釈されます。これがデフォルトです。 • オン – 値は、バイト数として解釈されます。 例 – 次のような特性のソース・ファイルにバイナリ **0x61 62 63 d6 d0 ce c4 61 62 63 64 65** が含まれているとします。 • ファイル・タイプ – 固定長 (可変行) • エンコーディング – GB2312 • ロー・デリミタ – '¥n' • カラム定義 – column1: 1-7; column 2: 9-10 [Fixed by Bytes] オプションを選択した場合 : • カラム 1 には、最初の 7 バイト (バイナリは **0x61 62 63 d6 d0 ce c4**) が表示されます。 • カラム 2 には、9 番目のバイトと 10 番目のバイト (バイナリは **0x62 63**) が表示されます。 [Fixed by Bytes] オプションを選択しなかった場合 : • カラム 1 には、最初の 7 文字 (バイナリは **0x61 62 63 d6d0 cec4 61 62**) が表示されます。 • カラム 2 には、それ以降の 2 文字 (バイナリは **0x64 65**) が表示されます。 **注意** 0xd6d0, c4c4 は、GB2312 では中国語の 2 文字を表します。
[Encoding]	ドロップダウン・メニューから適切な値を選択して、現在の文字コードを設定します。
[Column Header]	カラム名をファイルに書き込むオプションを選択します。

プロパティ	説明
[Header]	[Header] アイコンをクリックし、ファイルに書き込むレポート・ヘッダを作成します。Text Data Sink は、受信データの前にヘッダを書き込みます。これは、オプションの出力プロパティです。 書き込むヘッダにヘッダ・テキストを入力します。角カッコ表記では式を使用できます。
[Append Data]	受信データを送信先ファイルに追加するオプションを選択します。この値を設定しないと、Text Data Sink は送信先ファイルの既存データを上書きします。
[Write Block Size]	1 回の書き込み操作で Sybase ETL がファイルに書き込むレコードの数を指定します。

Text Data Sink デモ

Sybase ETL には、Text Data Sink コンポーネントのデモが複数含まれています。これらのデモは、[Help] メニューの Flash デモ、および DemoRepository のサンプル・プロジェクトとして使用できます。

Flash デモ

Flash デモを実行するには、[Help] - [Component Demonstrations] - [Destination] の順にクリックし、[Text Data Sink] を選択します。

DemoRepository

ナビゲータで、[Repository] - [TRANSFORMER.transformer.Repository] - [Projects] をクリックして、次を選択します。

- [Demo XML via SQL Data Provider]
- [Demo Text Data Sink Delimited/Fixed]

DB Bulk Load Sybase IQ

DB Bulk Load Sybase IQ は、Sybase IQ テーブルでバルク操作を実行する Destination コンポーネントです。このコンポーネントを使用すると、コンポーネントの IN ポートからのレコードに基づいて、Sybase IQ データベースのテーブルのレコードを操作できます。

DB Bulk Load Sybase IQ コンポーネントの設定

1 DB Bulk Load Sybase IQ を設計ウィンドウにドラッグします。

2 DB Bulk Load Sybase IQ の IN ポートをインバウンド・データを提供するコンポーネントの OUT ポートに接続します。

3 [Database Configuration] ウィンドウで、Sybase IQ 接続パラメータを追加します。

4 [Destination] アイコンをクリックして、インバウンド・データをロードするテーブルを選択します。新しい送信先テーブルに書き込む場合は、「新しい Sybase IQ 送信先テーブルの追加」(189 ページ) を参照してください。

5 [Load Stage] アイコンをクリックします。次のことができます。

- テンポラリ・データ・ファイルとして使用するファイル名を選択または入力して、[Save] をクリックします。

  – または –

- [Load Stage] フィールドにパイプ名を追加します (構文：`pipe://<name>`)。

  特定のフィールドの要件については、「DB Bulk Load Sybase IQ プロパティ・リスト」(197 ページ) を参照してください。

6 [Finish] をクリックします。

DB Bulk Load IQ をプロジェクトに追加するには、次を確認します。

- DB Bulk Load IQ をプロジェクトに追加する前に、Sybase IQ を実行しておく必要があります。ETL サーバおよび IQ データベースを同じマシン上で実行すると、パフォーマンスが向上します。これは必須ではありません。
- ターゲット・データベースを IQ に接続するには、有効なホスト名およびインタフェースを選択する必要があります。特定のフィールドの要件については、「DB Bulk Load Sybase IQ プロパティ・リスト」(197 ページ) を参照してください。

- データを新しい IQ テーブルにロードするには、DB Bulk Load の IN ポートまたはプロジェクトで使用可能なポートの構造に基づいて、送信先テーブルを作成できます。詳細については、「新しい Sybase IQ 送信先テーブルの追加」(189 ページ) を参照してください。
- スクリプトをカスタマイズするには、DB Bulk Load IQ コンポーネントを右クリックし、[Generate Load Script] を選択します。プロパティ・ウィンドウの [Load Script] アイコンをクリックして、スクリプトを編集および保存します。

❖ **ポート構造の更新**

データベースが変更されたポート構造を更新するには、次の手順に従います。

1　DB Bulk Load IQ コンポーネントを右クリックします。

2　[Reconfigure] を選択します。

[Reconfigure] オプションを選択すると、データベース・スキーマが変更されている場合、コンポーネントの設定が更新されます。現在の接続が閉じ、データベースへの新しい接続が開かれ、クエリのメタデータが読み取られ、ポート構造の更新が適用されます。

新しい Sybase IQ 送信先テーブルの追加

インバウンド・データを既存のテーブルに書き込むか、プロジェクトの既存のポートに基づいて新しい送信先テーブルを追加できます。

❖ **入力からの送信先テーブルの追加**

DB Bulk Load Sybase IQ の IN ポートに基づいて IQ 送信先テーブルを作成するには、次の手順に従います。

1　DB Bulk Load Sybase IQ コンポーネントを右クリックして、[Add Destination Table from Input] を選択します。

2　新しいテーブルの名前を入力します。

3　[OK] をクリックします。

❖ **既存のポートからの送信先テーブルの追加**

使用可能なコンポーネント・ポートに基づいて IQ 送信先テーブルを作成するには、次の手順に従います。

1 DB Bulk Load Sybase IQ コンポーネントを右クリックして、[Add destination table from port] を選択します。

2 新しいテーブルの名前を入力し、[OK] をクリックします。

3 テーブルの作成に使用するポートを選択します。[Apply] をクリックします。

クライアント側ロード・サポートの有効化

このコンポーネントを使用すると、Sybase IQ 以外のホスト・マシンにあるファイルから Sybase IQ テーブルにデータを追加できます。同じマシンに Sybase ETL と Sybase IQ をインストールする必要はありません。ETL サーバと Sybase IQ がネットワーク環境で通信するため、1 つのステップでリモート・マシンからのバルク・ロードが可能です。クライアント側をサポートするには、次の手順に従います。

- ETL サーバがインストールされているマシンに Sybase IQ 15 クライアントをインストールします。
- ETL Development と ETL サーバがインストールされているマシンに Sybase Adaptive Server Anywhere (ASA) 11 ODBC ドライバをインストールします。
- ターゲット IQ データベースのバージョンは Sybase IQ 15 である必要があります。
- Sybase IQ 15.0 サーバの [allow_read_client_file] オプションと [allow_write_client_file] オプションを有効にします。オプションは両方とも、IQ サーバごとに 1 回ずつ設定します。オプションを設定するには、次の手順に従います。

  a Sybase Central に移動し、Sybase IQ 15.0 サーバに接続します。

  b Sybase IQ サーバのデータベース名を右クリックし、[Options] を選択します。データベース・オプションのリストが表示されます。

c　[allow_read_client_file] および [allow_write_client_file] オプションを選択して、値を [On] に変更します。デフォルト値は [Off] です。

d　isql ユーティリティまたは dbisql ユーティリティを使用して、allow_read_client_file サーバ・オプションのプロパティを有効にします。

```
set option allow_read_client_file=on
GRANT READCLIENTFILE TO <group | user>
```

これらの手順を完了したら、コンポーネントのプロパティ・ウィンドウで [Use IQ Client Side Load] オプションを選択します。また、インタフェースとして ODBC を選択しないと、リモート・ホスト・マシンからのデータ・ロード・エラーが発生する場合があります。

注意 使用する ODBC ドライバが IQ 15 ODBC ドライバの場合、クライアント側ロードは ODBC でのみ機能します。

データをロードするための IQ の複数のライタの設定

Sybase ETL は、Sybase IQ 15 の複数の書き込み機能をサポートしています。この機能を使用すると、データを IQ データベースにロードするために複数のライタを追加できます。複数のライタによって、Sybase IQ テーブルの並列ロードが可能になり、逐次ロードよりも処理時間が短縮されます。複数のライタは、次のシナリオで使用できます。

- ソース・データベースで複数のテーブルを選択し、ターゲット IQ データベースの複数のテーブルに移行する場合。
- 複数のテーブルが含まれているマルチプロジェクト・コンポーネントからジョブを作成しているか、ジョブを実行するために並列実行トポロジにリンクされている複数のプロジェクトを選択している場合。

複数の書き込み機能を使用するには、次のターゲット IQ データベースのパーミッションを付与される必要があります。

オブジェクト名	種類	必要なパーミッション
ETL_MULTIPLEX_STATE	テーブル	create
ETL_MULTIPLEX_VERSION	テーブル	create
sp_iqstatistic	ストアド・プロシージャ	execute

注意 ETL は、IQ データベースに ETL_MULTIPLEX_STATE と ETL_MULTIPLEX_VERSION の 2 つのテーブルを作成します。ETL_MULTIPLEX_STATE テーブルの各ローは、ETL GridNode で選択された IQ ライタを示します。ライタは、各実行が終了すると自動的に削除されます。予期しないエラーによって GridNode がクラッシュした場合、ユーザがこのテーブルのデータを手動でクリーンにする必要があります。

Sybase Central を使用して必要なパーミッションを設定するには、次の手順に従います。

1 Sybase Central に移動し、Sybase IQ 15.0 サーバに接続します。

2 [Users & Groups] を展開し、テーブルの作成パーミッションを設定するユーザを選択します。

3 ユーザを右クリックし、[Properties] を選択します。

4 [Authorities] タブを選択し、[Resource] オプションをオンにし、ユーザ・パーミッションを付与して、IQ データベースにデータベース・オブジェクトを作成します。

5 [Permissions] タブを選択し、[Procedures & Functions] オプションを選択して、使用可能なすべてのパーミッションのリストを確認します。

6 sp_iqstatistics を選択し、対応する Execute カラムをクリックしてユーザ・パーミッションを付与し、IQ データベースでストアド・プロシージャを実行します。

7 [OK] をクリックして設定を保存します。

注意 マルチプレックス実行をサポートするには、ETL Development と ETL サーバがインストールされているマシンに Sybase Adaptive Server Anywhere (ASA) 11 ODBC ドライバをインストールする必要があります。

マルチプレックス実行に使用するライタは、*IQMultiplex.ini* ファイルで定義して設定する必要があります。

❖ *IQMultiplex.ini* ファイルでの目的のライタの定義

1　インストール・フォルダの *etc* ディレクトリに移動し、テキスト・エディタを使用して *IQMultiplex.ini* ファイルを開きます。

2　使用するマルチプレックス・グループごとに 1 つのセクションを追加します。デフォルトでは、*IQMultiplex.ini* ファイルは空です。特に指定がない場合、グリッド・エンジンは内部デフォルト値を使用します。

セクションのサンプルを次に示します。

```
[dbsybase15+iq15m+sample]
/* This is the section name */
Enabled=true
Workload=OperationsWaiting
MinimalUpdateInterval=60
SelectWriterTimeout=6
MostIdleNode=select NAME from TEST_CUSTOMER_NODE
order by WORKLOAD asc
MostBusyNode=select NAME from TEST_CUSTOMER_NODE
order by WORKLOAD desc
```

セクション名は、コーディネータのインタフェース名、ホスト名、およびデータベース名をプラス (+) 記号で区切って指定します。コロン (:)、ハッシュ・マーク (#)、または等号 (=) は使用できません。各グループの他のプロパティはオプションです。詳細については、表 5-1 (194 ページ) を参照してください。

3　ファイルを保存して閉じます。

表 5-1：マルチプレックス・グループのオプション・プロパティ

名前	種類	値	説明
Enabled	boolean	True (デフォルト) または False。	IQ サーバ・データベースでこの機能を有効または無効にします。
Workload	text	OperationsWaiting (デフォルト)。	負荷として使用する EXEC sp_iqstatistics の結果のローを指定します。 ディスパッチャによって、ストアド・プロシージャ EXEC sp_iqstatistics が実行され、ライタの負荷が問い合わせられます。クエリの結果セットによって、オペレーション・ステータス名が 2 番目のカラムに、ステータス値が 4 番目のカラムに返されます。 ディスパッチャは、指定する [Workload] オプションの値と一致する 2 番目のカラムのローを検索し、そのローの 4 番目のカラムを最終的な負荷値として使用します。
MinimalUpdateInterval	integer	ゼロ (0) より大きい整数。デフォルト値は 6 です。	ディスパッチャがコーディネータに問い合わせてライタの情報をリフレッシュする最小間隔 (単位は秒)。
SelectWriterTimeout	integer	ゼロ (0) 以上の整数。デフォルト値は 0 です。	すべてのライタが選択され、解放されていない場合、ディスパッチャが待機する秒数。0 を指定すると、ディスパッチャはいつまでも待機します。タイムアウトになると、エラーが生成されます。

名前	種類	値	説明

MostIdleNode	SQL	デフォルトでは、空です。	SQL 実行時に返される最初のローの最初のカラムは、ライタのノード名です。ディスパッチャは、返されたライタをマルチプレックスで最もアイドル状態が長いノードと見なし、次のテーブル・ロード要求のターゲットとして使用します。
			たとえば、この SQL クエリによって次のようにカスタムのディスパッチ・テーブルが作成されます。
			`DROP TABLE TEST_CUSTOMER_NODE;`
			`CREATE TABLE TEST_CUSTOMER_NODE(`
			`NAME varchar(100),`
			`WORKLOAD int /*must be integer*/`
			`);`
			`INSERT INTO TEST_CUSTOMER_NODE (NAME,WORKLOAD)VALUES('iq15w1',78);`
			`INSERT INTO TEST_CUSTOMER_NODE (NAME,WORKLOAD)VALUES('iq15w2',34);`
			`INSERT INTO TEST_CUSTOMER_NODE (NAME,WORKLOAD)VALUES('iq15w3',12);`
			`/* iq15w1-w3 are writers*/`
			テーブルから最もアイドル状態が長いノードを取得するには、*IQMultiplex.ini* ファイルにこの SQL クエリを追加します。
			`Select NAME from TEST_CUSTOMER_NODE order by WORKLOAD asc`
			iq15w3 が最もアイドル状態が長いノードとして返されます。

名前	種類	値	説明

MostBusyNode	SQL	デフォルトでは、空です。	SQL 実行時に返される最初のローの最初のカラムは、ライタのノード名です。ディスパッチャは、返されたライタをマルチプレックスの最もビジーなノードと見なし、ロード・テーブル・ターゲットとしての使用を表示します。
			たとえば、この SQL クエリによって次のようにカスタムのディスパッチ・テーブルが作成されます。
			`DROP TABLE TEST_CUSTOMER_NODE;`
			`CREATE TABLE TEST_CUSTOMER_NODE(`
			`NAME varchar(100),`
			`WORKLOAD int /*must be integer*/`
			`);`
			`INSERT INTO TEST_CUSTOMER_NODE (NAME,WORKLOAD)VALUES('iq15w1',78);`
			`INSERT INTO TEST_CUSTOMER_NODE (NAME,WORKLOAD)VALUES('iq15w2',34);`
			`INSERT INTO TEST_CUSTOMER_NODE (NAME,WORKLOAD)VALUES('iq15w3',12);`
			`/* iq15w1-w3 are writers*/`
			カスタムのディスパッチ・テーブルから最もビジーなノードを取得するには、*IQMultiplex.ini* ファイルにこの SQL クエリを追加します。
			`Select NAME from TEST_CUSTOMER_NODE order by WORKLOAD desc`
			iq15w1 が最もビジーなノードとして返されます。

DB Bulk Load Sybase IQ プロパティ・リスト

DB Bulk Load Sybase IQ プロパティ・リストは、[Database Configuration] ウィンドウで定義される接続パラメータおよび他のアイテムを識別します。

必須プロパティ

プロパティ	説明
[Interface]	データ・ソースへの接続に使用するメソッドまたはドライバを指定します。
[Host Name]	Sybase IQ ターゲットが実行されているホストを指定します。
[Destination]	既存のテーブルのセットからロード先テーブルを選択する場合、[Destination Table] アイコンをクリックします。
[Load Stage]	データ・ファイル・パスまたはパイプ名を指定します。load stage ファイルは、IQ サーバと同じマシン上に存在する必要があります。 UNIX または Linux 上で名前付きパイプを使用する場合、ETL サーバおよび IQ サーバは、同じマシン上に存在する必要があります。Windows ではオプションです。 **注意**　[Use IQ Client Side Load] オプションを選択した場合、名前付きパイプはサポートされません。

オプション・プロパティ

プロパティ	説明
[User and Password]	認証されたデータベース・ユーザとパスワードを指定し、認証されていないアクセスからデータベースを保護します。
[Key]	ターゲット・キー属性を選択し、選択した関数のレコードを特定します。 キーが選択されていない場合、インタフェースは、DB ホストから配信されるプライマリ・キー情報で動作します。使用できるプライマリ・キー情報がない場合、エラーが表示されます。

プロパティ	説明
関数	次のいずれかのロード関数を選択します。 • Insert (デフォルト) – 指定したファイル・パスまたはパイプ名を使用して、選択したターゲット・テーブルにレコードを直接ロードします。 • Upsert – 既存のレコードを更新し、新しいレコードを挿入します。Upsert を選択すると、既存のレコードは置換され、属性レベルで更新されません。[Key] プロパティを使用すると、ターゲットの属性を指定して、更新するレコードを指定できます。 次の場合、Upsert 関数は無視されます。 • [Generate Load Script] を使用している場合。LOAD TABLE スクリプトは、Insert 関数で使用するために生成されます。 • [Truncate] オプションを選択した場合。すべてのレコードは、データのロード前にターゲット・テーブルから削除されます。コンポーネントは、Upsert 関数を無視し、代わりに Insert 関数を実行して、すべての属性をターゲット・テーブルにロードします。 • Delete – 受信データのキーに基づいて、ターゲット・テーブルからレコードを削除します。[Key] プロパティを使用すると、ターゲットの属性を指定して、削除するレコードを指定できます。 Delete 関数と [Truncate] オプションを選択した場合、前処理 SQL 文と後処理 SQL 文のみが実行されます。
[Truncate]	ロード前に送信先テーブルからすべてのレコードを削除するオプションを選択します。
[Use IQ Client Side Load]	LOAD TABLE 文を使用して、リモート・ホスト・マシンにあるファイルから Sybase IQ テーブルにレコードを追加する場合は、このオプションを選択します。

プロパティ	説明
[Load Script]	プロパティが空の場合、LOAD TABLE 文はコンポーネント設定に基づいて実行時に生成されます。 カスタマイズされたスクリプトを使用するには、コンポーネントを右クリックして [Generate Load Script] を選択します。スクリプトの生成後、[Load Script] をクリックしてスクリプトを編集できます。
[Load Stage (Server)]	データ・ファイルへのサーバ・パスを指定するか、パイプを使用する際に空のままにしておきます。 Sybase IQ サーバが、[Load Stage] プロパティで指定したパス以外のテンポラリ・データ・ファイルへのパスを使用する必要がある場合、ここで入力する必要があります。
[Write Block Size]	1 回の書き込み操作でファイルまたはパイプに書き込まれるレコードの数を指定します。
[Pre-processing SQL]	コンポーネントの初期化時に実行するクエリを作成する場合、[Pre-processing SQL] アイコンをクリックします。 クエリには 1 つまたは複数の SQL 文を含めることができます。複数の文を使用する場合は、セミコロン (;) で区切ります。
[Post-Processing SQL]	すべてのコンポーネントの実行後に実行するクエリを作成する場合、[Post-processing SQL] アイコンをクリックします。 クエリには 1 つまたは複数の SQL 文を含めることができます。複数の文を使用する場合は、セミコロン (;) で区切ります。
[Database]	データ・ソースとして使用するデータベースを指定します。さらに、適切なインタフェース、場合によっては適切なユーザ ID およびパスワードを選択する必要があります。
[Schema]	データ・ソースとして使用するスキーマ／所有者を識別します。表示されたオブジェクトは適切に制限され、新しいテーブルがそのスキーマに作成されます。

プロパティ	説明
[Standardize Data Format]	異なるフォーマットをサポートするシステム間で Sybase ETL が移動できるように、受信する DATE および NUMBER 情報を標準フォーマットに変換するオプションを選択します。 日付は、年、月、日、時、分、秒、秒の小数点以下が含まれるフォーマット YYYY-MM-DD hh:mm:ss.s に変換されます。 例を示します。 2005-12-01 16:40:59.123 数値は、小数点として「.」を使用して変換されます。
[Database Options]	[Database Options] アイコンをクリックして、パフォーマンスのデフォルトを上書きし、一部のトランザクションの動作を制御するオプションを設定します。 詳細については、「データベース接続の設定」(105 ページ) を参照してください。
[IQ Lock Table in Exclusive Mode]	ターゲット・テーブルをロックし、同時トランザクションによる更新を防ぐオプションを選択します。排他テーブル・ロックが適用されると、他のトランザクションはロック・テーブルに対してクエリの発行またはあらゆる修正を実行できません。 また、[IQ Lock Table in Exclusive Mode] オプションによって、Sybase IQ 内の同じテーブルをロードする複数のプロジェクトがキューに入れられます。
[Wait Time for IQ Lock Table]	Exclusive ロックを取得する前に、プロジェクトが待機する最長ブロック時間を指定します。 time 引数は、hh:nn:ss.sss の形式で指定します。time 引数を入力しないと、Exclusive ロックが使用できるようになるか、または割り込みが発生するまで、サーバはいつまでも待機します。「00:00:00.000」を time 引数に指定すると、プロジェクトの開始時に Exclusive ロックが取得されます。
[Use IQ Multiplex]	複数のライタを使用してデータを IQ データベースにロードして、マルチプレックス実行をサポートする場合は、このオプションを選択します。

DB Bulk Load Sybase IQ および DB Space

Bulk Load Sybase IQ コンポーネントを使用する際、プロジェクトまたはジョブの実行に時間がかかる場合は、Sybase IQ コンソールまたはログをチェックします。「out of space」というメッセージが表示されている場合は、dbspace を追加する必要があります。IQ メッセージ・ファイルのメッセージには、領域が不足している dbspace と、最小限追加すべき領域のサイズ (MB 単位) が示されます。データの挿入時に問題が起きる場合は、IQ ストアの領域を増やす必要があると考えられます。大量のソートおよびマージを行うクエリ時に問題が起きる場合は、テンポラリ・ストアの領域を増やす必要があると考えられます。

この SQL 文は、100MB のデータベース領域を Windows の既存の asiqdemo データベースに追加します。

```
CREATE DBSPACE asiqdemo2 AS
'd:¥¥sybase¥¥ASIQ-12_7¥¥demo¥¥asiqdemo2.iq'
IQ STORE
SIZE 100;
```

この SQL 文は、200MB のテンポラリ領域を既存の asiqdemo データベースに追加します。

```
CREATE DBSPACE asiqdemotmp AS
'd:¥¥sybase¥¥ASIQ-12_7¥¥demo¥¥asiqdemo2.iqtmp'
IQ TEMPORARY STORE
SIZE 200 ;
```

注意 潜在的なメモリの問題の診断に関する詳細については、『Sybase IQ トラブルシューティングおよびリカバリ・ガイド』のリソースの問題を参照してください。

IQ Loader のデータ・フォーマットのカスタマイズ

データ・ファイルまたはパイプに書き込む際、および LOAD TABLE スクリプトを再生する際、IQ Loader インタフェースは、デリミタ、null 処理および文字セットのデフォルト値を使用します。デフォルト値は、ETL サーバの *INI* ファイルを次のように指定できます。

グループ	キー	値	デフォルト	説明
iq_loader	rowdelim	任意の文字列。改行の代わりに '¥n' を使用します。	' ¥n '	行デリミタ
iq_loader	coldelim	任意の文字列。タブの代わりに '¥t' を使用します。	' |@#& '	カラム・デリミタ
iq_loader	nullreplace	任意の文字列。空の場合、NULL 句は Load Script に追加されません。	' [NULL] '	NULL 値に使用される文字列
iq_loader	characterset	IQ によってサポートされる任意の文字セット。	' ' (=auto)	データ・ファイルに使用されるエンコード

Loader コンポーネント

Loader コンポーネントは、変換を実行せずに、ソース・データベースまたはファイルから IQ データベースにデータをロードするのに役立ちます。

コンポーネント	説明
IQ Loader File via Load Table	このコンポーネントは、LOAD TABLE 文を使用してファイルからターゲット IQ データベースにデータをロードする場合に使用します。
IQ Loader DB via Insert Location	このコンポーネントは、INSERT LOCATION 文を使用してソース・データベースからターゲット IQ データベースにデータをロードする場合に使用します。

IQ Loader File via Load Table

IQ Loader File via Load Table コンポーネントは、自動的に生成された LOAD TABLE 文を使用して、ファイルからターゲット IQ データベースにデータをロードする場合に使用します。

このコンポーネントは、自己完結型のコンポーネントで、ソース・ファイルから読み取る場合はデータ・ソースとして、Sybase IQ データベースに書き込む場合はデータ・シンクとして機能します。このコンポーネントは、入出力ポートを持ちません。そのため、ソース・ファイルからの読み取りを実行するためのコンポーネントや、Sybase IQ で Load Table を呼び出すためのコンポーネントを作成する必要はありません。ETL では、区切り付きテキスト・ファイルからデータを抽出する Load Table 文が自動的に生成され、Sybase IQ にデータがロードされます。

IQ Loader File via Load Table コンポーネントの設定

1. IQ Loader File via Load Table を設計ウィンドウにドラッグします。
2. [Database Configuration] ウィンドウで、ターゲット IQ データベースの接続パラメータを追加します。特定のフィールドの要件については、「IQ Loader DB via Insert Location プロパティ・リスト」(214 ページ) を参照してください。
3. ロード先テーブルを選択または入力します。
4. [Finish] をクリックします。
5. プロパティ・ウィンドウで他のオプション・プロパティを指定します。

[Text Source] プロパティ・ウィンドウの操作

[Text Source] プロパティ・ウィンドウを使用して、ソース・ファイルのデータの構造プロパティを定義できます。このファイルは以下のパネルで構成されます。

- [File Content] パネル – ソース・ドキュメントの内容を表示します。
- [Properties] パネル – ファイルの説明プロパティを表示します。
- [Preview] パネル – 現在選択されているプロパティに基づいて、ソース・ファイルのデータのテーブル・ビューを表示します。

クライアント側ロード・サポートの有効化

このコンポーネントを使用すると、リモート・ホスト・マシンにあるファイルから Sybase IQ テーブルにデータをロードできます。クライアント側ロード・サポートを有効にする方法については、「クライアント側ロード・サポートの有効化」(190 ページ) を参照してください。

データをロードするための IQ の複数のライタの設定

データを IQ にロードするために複数のライタを使用してマルチプレックス実行のサポートを有効にするには、追加の設定が必要です。設定手順の詳細については、「データをロードするための IQ の複数のライタの設定」(191 ページ) を参照してください。

IQ Loader File via Load Table プロパティ・リスト

次の表は、IQ Loader File via Load Table コンポーネントの必須プロパティとオプション・プロパティの一覧です。

必須プロパティ

プロパティ	説明
[Interface]	ターゲット IQ データベースへの接続に使用するメソッドまたはドライバを指定します。サポートされているインタフェースは Sybase および ODBC です。
[Host Name]	Sybase IQ ターゲットが実行されているホストを指定します。
[Destination]	既存のテーブルのセットからロード先テーブルを選択する場合にクリックします。
[Text Source]	データ・ソースとして使用するテキスト・ファイルを識別します。プロパティ・ウィンドウで、[Text Source] アイコンをクリックし、ファイルを選択して、形式を指定します。詳細については、「[Text Source] プロパティ・ウィンドウの操作」(204 ページ) を参照してください。
[Row Delimiter]	各ローを区切る方法を指定します。 • [LF] (改行) • [CR] (行頭復帰) • [CRLF] (行頭復帰とそれに続く改行) 別のデリミタ文字を使用することもできます。
[Column Delimiter]	カラムを区切る方法を指定します。 • [Tab] • [Comma] • [Semicolon] 別のデリミタ文字を使用することもできます。

オプション・プロパティ

プロパティ	説明
[User and Password]	認証されたデータベース・ユーザとパスワードを指定し、認証されていないアクセスからデータベースを保護します。
[Key]	ターゲット・キー属性を選択し、選択した関数のレコードを特定します。 キーが選択されていない場合、インタフェースは、DB ホストから配信されるプライマリ・キー情報で動作します。使用できるプライマリ・キー情報がない場合、エラーが表示されます。
[Function]	次のいずれかのロード関数を選択します。 • Insert – ファイルから選択したターゲット・テーブルにレコードを直接ロードします。 • Upsert – 既存のレコードを更新し、新しいレコードを挿入します。Upsert を選択すると、既存のレコードは置換され、属性レベルで更新されません。[Key] プロパティを使用すると、ターゲットの属性を指定して、更新するレコードを指定できます。 次の場合、Upsert 関数は無視されます。 • [Generate Load Script] を使用している場合。LOAD TABLE スクリプトは、Insert 関数で使用するために生成されます。 • [Truncate] オプションを選択した場合。すべてのレコードは、データのロード前にターゲット・テーブルから削除されます。コンポーネントは、Upsert 関数を無視し、代わりに Insert 関数を実行して、すべての属性をターゲット・テーブルにロードします。
[Use IQ Client Side Load]	LOAD TABLE 文を使用して、リモート・ホスト・マシンにあるファイルからターゲット IQ データベースにデータをバルク・ロードする場合は、このオプションを選択します。
[Load Script]	プロパティが空の場合、Load Table 文はコンポーネント設定に基づいて実行時に生成されます。 カスタマイズされたスクリプトを使用するには、コンポーネントを右クリックして [Generate Load Script] を選択します。スクリプトの生成後、[Load Script] をクリックしてスクリプトを編集できます。

プロパティ	説明
[Truncate]	ロード前に送信先テーブルからすべてのレコードを削除するオプションを選択します。
[Pre-processing SQL]	コンポーネントの初期化時に実行するクエリを作成する場合、[Pre-processing SQL] アイコンをクリックします。 クエリには 1 つまたは複数の SQL 文を含めることができます。複数の文を使用する場合は、セミコロン (;) で区切ります。
[Post-Processing SQL]	すべてのコンポーネントの実行後に実行するクエリを作成する場合、[Post-processing SQL] アイコンをクリックします。 クエリには 1 つまたは複数の SQL 文を含めることができます。複数の文を使用する場合は、セミコロン (;) で区切ります。
[Database]	データ・ターゲットとして使用するデータベースを指定します。データベースは、指定したユーザ名、パスワード、およびホスト名とともに使用されます。
[Schema]	テーブル・カタログをフィルタする所有者を指定します。
[Database Options]	[Database Options] アイコンをクリックして、パフォーマンスのデフォルトを上書きし、一部のトランザクションの動作を制御するオプションを設定します。 詳細については、「データベース接続の設定」(105 ページ) を参照してください。
[Null Indicator]	ソース・ファイルで null 値を表す文字列を指定します。
[Skip Rows]	ロード処理に対して、入力ファイルの開始時に省略するローの数を指定します。デフォルトは 0 です。
[Parallel format]	最後のカラムを含むすべてのカラムが 1 つの ASCII 文字で区切られている場合、このオプションを選択すると、LOAD TABLE コマンドを並列で実行できます。
[Strip]	このオプションは、値が挿入される前に後続ブランクを値から削除する場合に選択します。このプロパティは、可変長非バイナリ・データにのみ適用されます。

プロパティ	説明
[Byte Order]	読み込み時のバイトの順序を指定します。このオプションはすべてのバイナリ入力フィールドに適用します。何も定義されなければ、このオプションは無視されます。Sybase ETL は必ずバイナリ・データを、自分が動作しているコンピュータのネイティブ・フォーマットで読み込みます (デフォルトは NATIVE です)。または、次のように指定できます。 • HIGH。マルチバイトの値が上位バイト優先である場合に指定します。 • LOW。マルチバイトの値が下位バイト優先である場合に指定します。
[Block Size]	入力を読み込むデフォルト・サイズをバイト数で指定します。
[Limit]	テーブルに挿入するローの最大数を指定します。制限なしのデフォルトは 0 です。
[ON File Error]	入力ファイルが存在しないか、またはファイルを読み込むパーミッションが不正であるためにファイルを開くことができない場合の Sybase IQ の動作を指定します。その他の理由やエラーによる場合は、挿入処理全体がアボートします。次のオプションのいずれかを指定できます。 • ROLLBACK。トランザクション全体をアボートします (デフォルト)。 • FINISH。すでに完了している挿入処理を完了して、ロード処理を終了します。 • CONTINUE。エラーを返しますが、該当するファイルのみを省略してロード処理を継続します。このオプションは部分幅挿入とともに使用することはできません。
[Word Skip]	ワード・インデックスの作成時に、指定された制限よりも長いデータがあった場合にロードを続行することができます。

プロパティ	説明
[IQ Lock Table in Exclusive Mode]	ターゲット・テーブルをロックし、同時トランザクションによる更新を防ぐオプションを選択します。排他テーブル・ロックが適用されると、他のトランザクションはロック・テーブルに対してクエリの発行またはあらゆる修正を実行できません。 また、[IQ Lock Table in Exclusive Mode] オプションによって、Sybase IQ 内の同じテーブルをロードする複数のプロジェクトがキューに入れられます。
[Wait Time for IQ Lock Table]	Exclusive ロックを取得する前に、プロジェクトが待機する最長ブロック時間を指定します。 time 引数は、hh:nn:ss.sss の形式で指定します。time 引数を入力しないと、Exclusive ロックが使用できるようになるか、または割り込みが発生するまで、サーバはいつまでも待機します。「00:00:00.000」を time 引数に指定すると、プロジェクトの開始時に Exclusive ロックが取得されます。
[Use IQ Multiplex]	複数のライタを使用してデータを IQ データベースにロードして、マルチプレックス実行をサポートする場合は、このオプションを選択します。

IQ Loader DB via Insert Location

IQ Loader DB via Insert Location コンポーネントは、Insert Location 文を使用して、ソース・データベースからターゲット IQ データベースにデータをロードする場合に使用します。

このコンポーネントは、自己完結型のコンポーネントで、ソース・データベースから読み取る場合はデータ・ソースとして、Sybase IQ データベースに書き込む場合はデータ・シンクとして機能します。このコンポーネントは、入出力ポートを持ちません。そのため、ソース・ファイルからの読み取りを実行するためのコンポーネントや、Sybase IQ で Insert Location を呼び出すためのコンポーネントを作成する必要はありません。ETL では、Insert Location 文を自動的に生成して、データをソース・データベースから Sybase IQ に転送します。

- 12.0 より前のバージョンの Sybase IQ からバージョン 12.0 以降にカラムを移動できます。
- Adaptive Server® Enterprise または Sybase IQ から Sybase IQ への最適化されたロードが可能です。
- また、Sybase Enterprise Connect™ Data Access (ECDA) を使用して、Sybase Adaptive Server Enterprise、Oracle、IBM DB2、および Microsoft SQL Server から Sybase IQ をロードすることができます。ETL 4.8 は、IBM DB2 9.1、Oracle 10g、および Microsoft SQL Server 2005 での Sybase ECDA 15.0 の使用をサポートしています。

  Sybase ECDA のマニュアルについては、Sybase Product Manuals Web サイト (http://www.sybase.com/support/manuals) を参照してください。

IQ Loader DB via Insert Location コンポーネントの設定

1. IQ Loader DB via Insert Location コンポーネントを設計ウィンドウにドラッグします。
2. 送信先データベースの IQ データベース接続プロパティを入力します。
   - [Host] – IQ ホストを選択します。
   - [User] – 認証されたデータベース・ユーザ名を入力します。
   - [Password] – データベース・ユーザのパスワードを入力します。
   - [Database] – 送信先データベースとして使用するデータベースを選択します。
   - [Schema] – 表示されたオブジェクトを制限するスキーマまたは所有者を選択し、そのスキーマで新しいテーブルを作成します。
   - [Processing] をクリックして、IQ 送信先データベースの処理前の SQL と処理後の SQL を指定します。
   - 使用可能なテーブルのリストを表示するには、[Logon] をクリックします。
   - [Next] をクリックします。
3. ソース・データベースの接続情報を入力して、転送するテーブルを選択します。
   - ソース・データベース・オプションにアクセスするために [Use remote server definition] を選択して、ソースからデータおよびメタデータを取得します。このオプションは、Create Server コマンドを使用して送信先 IQ データベースのリモート・サーバとしてソース・サーバが定義されている場合にのみ選択できます。このオプションが選択されていない場合、ユーザは、*.INI* ファイルまたは *interfaces* ファイルの設定情報に従い、ソース・データに直接接続されます。
   - [Host] – データ・ソースを選択します。
   - [Database] – 使用するデータベースを選択します。
   - [Schema] – 表示されたオブジェクトを制限するスキーマまたは所有者を選択し、そのスキーマで新しいテーブルを作成します。
   - [Processing] をクリックして、ソース・データベースの処理前の SQL と 処理後の SQL を指定します。

- ロード先テーブルが存在しない場合は、[Create Target Tables] オプションを選択して、これを作成します。
- [Continue on Error] オプションは、データベースへのデータのロード中にエラーが発生しても処理を継続する場合に選択します。
- [Encrypted Password] オプションは、暗号化フォーマットでパスワードを送信する場合に選択します。

注意 リモート・サーバとして使用された場合、Sybase IQ はこのパスワード暗号化をサポートしません。

- [Use IQ Multiplex] オプションは、データを IQ にロードするために複数のライタを使用してマルチプレックス実行をサポートする場合に選択します。このオプションは、複数のテーブルが IQ データベースに移行中の場合に選択します。
- [Lock Table] オプションは、[Exclusive] モードでターゲット・テーブルをロックし、同時トランザクションによる更新を防ぐ場合に選択します。このオプションを選択すると、他のトランザクションはロック・テーブルに対してクエリの発行またはあらゆる修正を実行できません。また、[Lock Table] オプションによって、Sybase IQ 内の同じテーブルをロードする複数のプロジェクトがキューに入れられます。

  このオプションを選択する場合、ロックを取得する前に、プロジェクトが待機する最長ブロック時間も指定する必要があります。
- [Packet Size] フィールドにネットワーク・パケット・サイズを入力します。
- [Limit Rows] の値を入力します。
- [Skip Rows] フィールドで、ロード処理に対して入力テーブルの開始時に省略するローの数を指定します。
- 指定したデータベースで使用可能なテーブルのリストを表示するには、[Logon] をクリックします。デフォルトでは、各テーブルが転送用に選択されています。転送しないテーブルについては、[Transfer] オプションの選択を解除します。また、1 つまたは複数のテーブル・ローを選択して、右クリックし、[Exclude] を選択することもできます。テーブルを転送対象に含めるには、右クリックして [Transfer] を選択します。

あるいは、次の手順を実行できます。

- すべてのテーブルを除外する場合は、[Exclude all objects from transfer] アイコンをクリックします。
- すべてのテーブルを転送対象に含めるには、[Include all objects in transfer] アイコンをクリックします。

- [Next] をクリックします。

4 ソース・テーブルを確認します。ソース・テーブルは次のフォーマットで作成されている必要があります。

```
source_schema.source_table
```

5 ドロップダウン・メニューからロード先テーブルを選択します。ソースとロード先について、一対一のマッピング (ロード先ごとに 1 つのソース) が存在します。

6 ロード先テーブルからすべての既存のデータ・ローを削除する場合、[Truncate Destination] オプションをクリックします。

7 [Next] をクリックします。

8 ロード設定サマリを確認します。[Finish] をクリックします。

Insert Location 文のロケールの設定

ユーザが Insert Location 文を実行すると、Sybase IQ は言語を判断するために必要なローカライゼーション情報、照合順、文字セット、および日付と時刻の形式をロードします。データベースがプラットフォームのデフォルト以外のロケールを使用している場合は、ローカル・クライアントに環境変数を設定して、Sybase IQ が正しい情報をロードするようにしてください。

環境変数 LC_ALL を設定すると、Sybase IQ はその値をロケール名として使用します。LC_ALL が設定されていない場合は、Sybase IQ は LANG 環境変数の値を使用します。どちらの環境変数も設定されていない場合は、Sybase IQ はロケール・ファイルにあるデフォルトのエントリを使用します。例については、『Sybase IQ 12.7 システム管理ガイド』の「第 11 章 国際化言語と文字セット」の「ロケールの設定」を参照してください。

データをロードするための IQ の複数のライタの設定

データを IQ にロードするために複数のライタを使用してマルチプレックス実行のサポートを有効にするには、追加の設定が必要です。設定手順の詳細については、「データをロードするための IQ の複数のライタの設定」(191 ページ) を参照してください。

IQ Loader DB via Insert Location プロパティ・リスト

IQ Loader DB via Insert Location プロパティ・リストでは、接続パラメータ、および IQ Loader DB via Insert Location コンポーネント・ウィンドウで定義する必要がある他の項目を確認します。

必須プロパティ

プロパティ	説明
[IQ Host Name]	Sybase IQ ターゲットが実行されているホストを指定します。
[IQ User]	認証された IQ ユーザを指定して、認証されていないアクセスからデータベースを保護します。
[IQ Password]	パスワードを指定して、認証されていないアクセスからデータベースを保護します。
[Source Host Name]	データ・ソースを指定します。
[Source Database]	ソース・データベースを指定します。
[Source Transfer List]	スキーマの修飾されたソース・テーブル名とターゲット・テーブル名を指定します。ターゲット・トランケート・カラムで、ターゲット・テーブルをトランケートする場合は 1 を、それ以外の場合は 0 を指定します。

オプション・プロパティ

プロパティ	説明
[IQ Database]	IQ 送信先データベースを指定します。
[IQ Schema]	表示されたオブジェクトを制限する IQ スキーマまたは所有者を選択し、そのスキーマで新しいテーブルを作成します。
[IQ Pre-processing SQL]	コンポーネントの初期化時に実行するクエリを作成する場合、[IQ Pre-processing SQL] アイコンをクリックします。 クエリには 1 つまたは複数の SQL 文を含めることができます。複数の文を使用する場合は、セミコロン (;) で区切ります。

プロパティ	説明
[IQ Post-processing SQL]	[IQ Post-processing SQL] アイコンをクリックして、すべてのコンポーネントの実行後に実行するクエリを作成します。 クエリには 1 つまたは複数の SQL 文を含めることができます。複数の文を使用する場合は、セミコロン (;) で区切ります。
[Use Remote Definition]	このオプションは、Create Server コマンドを使用して送信先 IQ データベースのリモート・サーバとしてソース・サーバが定義されている場合にのみ選択できます。このオプションが選択されていない場合、ユーザは、*.INI* ファイルまたは *interfaces* ファイルの設定情報に従い、ソース・データに直接接続されます。
[Source Schema]	表示されたオブジェクトを制限するスキーマまたは所有者を指定します。
[Source Pre-processing SQL]	コンポーネントの初期化時に実行するクエリを作成する場合、[Source Pre-processing SQL] アイコンをクリックします。 クエリには 1 つまたは複数の SQL 文を含めることができます。複数の文を使用する場合は、セミコロン (;) で区切ります。
[Source Post-processing SQL]	[Source Post-processing SQL] アイコンをクリックして、すべてのコンポーネントの実行後に実行するクエリを作成します。 クエリには 1 つまたは複数の SQL 文を含めることができます。複数の文を使用する場合は、セミコロン (;) で区切ります。

プロパティ	説明
[Function]	次のいずれかのロード関数を選択します。 • Insert – ソースから選択したターゲット・テーブルにレコードを直接ロードします。 • Upsert – 既存のレコードを更新し、新しいレコードを挿入します。Upsert を選択すると、既存のレコードは置換され、属性レベルで更新されません。対象のテーブルには、事前に定義されたプライマリ・キーが含まれている必要があります。 次の場合、Upsert 関数は無視されます。 • [Generate Load Script] を使用している場合。INSERT LOCATION スクリプトは、Insert 関数で使用するために生成されます。 • [Truncate] オプションを選択した場合。すべてのレコードは、データのロード前にターゲット・テーブルから削除されます。コンポーネントは、Upsert 関数を無視し、代わりに Insert 関数を実行して、すべての属性をターゲット・テーブルにロードします。
[Create Target Tables]	ロード先テーブルが存在しない場合は、このオプションを選択して、これを作成します。
[Continue on Error]	このオプションは、データベースへのデータのロード中にエラーが発生しても、実行を継続する場合に選択します。
[Limit Rows]	テーブルに挿入するローの最大数を指定します。制限なしのデフォルトは 0 です。
[Skip Rows]	ロード処理に対して、入力テーブルの開始時に省略するローの数を指定します。デフォルトは 0 です。
[Encrypted Password]	このオプションは、暗号化フォーマットでパスワードを送信する場合に選択します。
[Packet Size]	ネットワーク・パケット・サイズを指定します。
[Load Script]	Insert Location 文は、プロパティが空の場合、コンポーネントの設定に基づいて実行時に生成されます。 カスタマイズされたスクリプトを使用するには、コンポーネントを右クリックして [Generate Load Script] を選択します。スクリプトの生成後、[Load Script] をクリックしてスクリプトを編集できます。

プロパティ	説明
[Use IQ Multiplex]	複数のライタを使用してデータを IQ データベースにロードして、マルチプレックス実行をサポートする場合は、このオプションを選択します。
[IQ Lock Table in Exclusive Mode]	ターゲット・テーブルをロックし、同時トランザクションによる更新を防ぐオプションを選択します。排他テーブル・ロックが適用されると、他のトランザクションはロック・テーブルに対してクエリの発行またはあらゆる修正を実行できません。 また、[IQ Lock Table in Exclusive Mode] オプションによって、Sybase IQ 内の同じテーブルをロードする複数のプロジェクトがキューに入れられます。
[Wait Time for IQ Lock Table]	Exclusive ロックを取得する前に、プロジェクトが待機する最長ブロック時間を指定します。 time 引数は、hh:nn:ss.sss の形式で指定します。time 引数を入力しないと、Exclusive ロックが使用できるようになるか、または割り込みが発生するまで、サーバはいつまでも待機します。「00:00:00.000」を time 引数に指定すると、プロジェクトの開始時に Exclusive ロックが取得されます。

Job コンポーネント

Job コンポーネントは、ジョブの実行を管理します。

コンポーネント	説明
Start	ジョブの開始を表します。Start はジョブに追加する最初のコンポーネントです。
Project	ジョブで実行するプロジェクトを特定します。このコンポーネントは、ジョブの個々のプロジェクトを実行する場合に使用します。
Synchronizer	ジョブ・フローの実行を管理します。このコンポーネントは、以前に実行されたプロジェクトのステータスに応じてジョブのフローを管理する場合に使用します。 各プロジェクトを重大なプロジェクトまたは重大でないプロジェクトとして定義できます。重大なプロジェクトのエラーは、Synchronizer のシグナル • エラーの原因となります。
Multi-Project	ジョブ内のプロジェクト・グループを視覚的に表します。Multi-Project によって、Project コンポーネントと Synchronizer コンポーネントのプロパティが組み合わされます。 このコンポーネントは、非常に多くの独立したプロジェクトでジョブが構成されている場合、つまり、任意の順番でプロジェクトが実行される可能性があるジョブの場合に使用します。
Finish	ジョブが正常に実行され、終了したことを表します。 このコンポーネントは、ジョブが正常に終了したことを示す場合に使用します。
Error	ジョブの実行中にエラーが発生し、正常に終了しなかったことを視覚的に表します。 このコンポーネントは、ジョブが正常に終了しなかったことを示す場合に使用します。

Start

Start はジョブに追加する最初のコンポーネントです。このコンポーネントをジョブに追加するには、[Component Store] から設計ウィンドウにドラッグします。

注意 複数の Project コンポーネント、Multi-Project コンポーネント、または両方を Start コンポーネントに接続できます。

Project

Project コンポーネントは、ジョブで実行するプロジェクトを特定します。このコンポーネントは、ジョブの個々のプロジェクトを実行する場合に使用します。

ジョブへの Project コンポーネントの追加

❖ **Project コンポーネントの追加および設定**

1. Project コンポーネントをジョブに追加するには、[Component Store] から設計ウィンドウにドラッグします。
2. 隣接するコンポーネントと接続します。
3. コンポーネントをダブルクリックし、実行するプロジェクトを選択します。

必須プロパティ

プロパティ	説明
[Project Name]	追加するプロジェクトを選択する場合、[Project Name] アイコンをクリックします。

オプション・プロパティ

プロパティ	説明
[Continue on DB Write Errors]	このオプションは、DB Data Sink コンポーネントを使用してデータベースにデータをロード中にエラーが発生しても、プロジェクトの実行を継続する場合に選択します。このオプションが原因でエラーが無視された場合、プロジェクトに「エラー」が示されます。このオプションは、Reject Log (「データベース接続の設定」(105 ページ) を参照) と一緒に使用して、拒否されたレコードを「後処理」できます。

Project コンポーネントのデモ

DemoRepository でサンプル・ジョブを確認します。サンプル・ジョブを実行するには、次の手順に従います。

1 ナビゲータで、[Repository] - [TRANSFORMER.transformer.Repository] - [Jobs] をクリックします。

2 以下の項目を選択します。

- [Demo Transfer all German Data]
- [Demo Transfer U.S. Sales on an incremental basis]

Synchronizer

Synchronizer は、ジョブのフローの実行を管理します。

このコンポーネントは、以前に実行されたプロジェクトのステータスに応じてジョブのフローを管理する場合に使用します。各プロジェクトを重大なプロジェクトまたは重大でないプロジェクトとして定義できます。重大なプロジェクトのエラーは、Synchronizer の Error ポートのブランチに沿ってジョブ・フローが進行する原因になります。

Synchronizer コンポーネントの Success ポートと Error ポートは、両方とも次のコンポーネントと接続可能です。

- Multiple Project コンポーネント。
- Multiple Multi-Project コンポーネント。
- Multiple Project コンポーネントと Multi-Project コンポーネント。
- Single Finish コンポーネントまたは Error コンポーネント。

❖ **Synchronizer コンポーネントの追加および設定**

1　コンポーネントをジョブに追加して、実行ステータスをこのコンポーネントに送信するすべてのプロジェクトに接続します。

2　(オプション) 重大なプロジェクトを選択する場合は、プロパティ・ウィンドウで [Synchronize Options] アイコンをクリックします。

Synchronizer コンポーネントのデモ

DemoRepository でサンプル・ジョブを確認します。サンプル・ジョブを実行するには、次の手順に従います。

1　ナビゲータで、[Repository] - [TRANSFORMER.transformer.Repository] - [Jobs] をクリックします。

2　[Demo Transfer all German Data] を選択します。

Multi-Project

Multi-Project は、ジョブ内のプロジェクト・グループを視覚的に表します。Multi-Project によって、Project コンポーネントと Synchronizer コンポーネントのプロパティが組み合わされます。Multi-Project コンポーネントの Success ポートと Error ポートは、両方とも次のコンポーネントと接続可能です。

- Multiple Project コンポーネント。
- Multiple Multi-Project コンポーネント。
- Multiple Project コンポーネントと Multi-Project コンポーネント。
- Single Finish コンポーネントまたは Error コンポーネント。

このコンポーネントは、非常に多くの独立したプロジェクトでジョブが構成されている場合、つまり、任意の順番でプロジェクトが実行される可能性がある (マルチエンジン・ジョブで使用するときは、並列実行される場合もある) ジョブの場合に使用します。

❖ Multi-Project コンポーネントの設定

1 ジョブにコンポーネントを追加して、隣接するコンポーネントと接続します。

2 プロパティ・ウィンドウで、[Projects Execution] アイコンをクリックし、実行するプロジェクトを選択します。

3 プロジェクトをグループに追加する場合は、ナビゲータでプロジェクト名を右クリックし、[Add Projects] を選択します。

4 グループからプロジェクトを削除する場合は、ナビゲータでプロジェクトを選択し、[Remove Projects] を選択します。

注意 プロジェクト・グループに含まれるプロジェクトのインスタンスは、1 つだけです。したがって、プロジェクトを複数回追加することはできません。

プロジェクトの実行に使用するオプションは次のとおりです。

- **[Continue on Error]** – このオプションは、Project コンポーネントの [Continue on DB Write Errors] プロパティに対応します。このオプションを選択すると、データベースへのデータのロード中にエラーが発生してもプロジェクトの実行が継続されます。
- **[Critical]** – 個々のプロジェクトを重大なプロジェクトまたは重大ではないプロジェクトとして定義できます。重大なプロジェクトでエラーが発生すると、Multi-Project コンポーネントがシグナル • エラーの原因となります。

Multi-Project コンポーネントのデモ

DemoRepository でサンプル・ジョブを確認します。サンプル・ジョブを実行するには、次の手順に従います。

1　ナビゲータで、[Repository] - [TRANSFORMER.transformer.Repository] - [Jobs] をクリックします。

2　[Demo Transfer all U.S. Data] を選択します。

Finish

Finish は、ジョブが正常に実行され終了したことを視覚的に表します。このコンポーネントは、ジョブが正常に終了したことを示す場合に使用します。Finish は、Synchronize、Project、または Multi-Project コンポーネントに接続できます。

Finish コンポーネントのデモ

DemoRepository でサンプル・ジョブを確認します。サンプル・ジョブを実行するには、次の手順に従います。

1　ナビゲータで、[Repository] - [TRANSFORMER.transformer.Repository] - [Jobs] をクリックします。

2　以下の項目を選択します。

- [Demo Transfer all German Data]
- [Demo Transfer all U.S. Data]
- [Demo Transfer U.S. Sales on an incremental basis]

Error

Error コンポーネントは、ジョブの実行中にエラーが発生し、正常に終了しなかったことを視覚的に表します。このコンポーネントは、ジョブが正常に終了しなかったことを示す場合に使用します。Error は、Synchronize、Project、または Multi-Project コンポーネントに接続できます。

Error コンポーネントのデモ

DemoRepository でサンプル・ジョブを確認します。サンプル・ジョブを実行するには、次の手順に従います。

1 ナビゲータで、[Repository] - [TRANSFORMER.transformer.Repository] - [Jobs] をクリックします。

2 以下の項目を選択します。

- [Demo Transfer all German Data]
- [Demo Transfer all U.S. Data]

第 6 章

Sybase ETL サーバ

この章では、Sybase ETL サーバの使用方法について説明します。詳細については、「Sybase ETL アーキテクチャ」(2 ページ) を参照してください。

Sybase ETL サーバは、スケーラブルな分散型のグリッド・エンジンです。データ・ソースに接続し、Sybase ETL Development で設計された変換フローを使用して、データ・ターゲットへのデータの抽出およびロードを行います。Sybase ETL サーバは UDP ブロードキャストを使用して、起動や停止、システム障害、クラッシュなどの緊急イベントを他のサーバに通知します。

通信用のデフォルト・ポートは 5124 です。デフォルト・ポートは、*INI* ファイルまたはコマンド・ラインで変更できます。

サーバ間のすべての通信は、同じポートで TCP/IP を介して行われます。

注意 このポートをブロックしているファイアウォールがないことと、このポートが使用中でないことを確認してください。必要に応じて、すべてのサーバ・インストールのポートを別の番号に変更することもできます。

Sybase ETL サーバの起動

コマンド・プロンプトで次のように入力します。

Windows の場合：

```
GridNode
GridNode --port 5500
```

Linux および UNIX の場合：

```
GridNode.sh
GridNode.sh --port 5500
```

ETL サーバの停止

ローカル・プロセスまたはリモート・プロセスの場合、コンソールからサーバを停止できます。サーバが停止する前に、現在実行されているすべてのプロジェクトの実行が完了します。ETL サーバを停止するには、コマンド・プロンプトで次のように入力します。

Windows の場合：

```
GridNode --shutdown
GridNode --shutdown --server [remotehost] --port [port]
```

Linux および UNIX の場合：

```
GridNode.sh --shutdown
GridNode.sh --shutdown --server[remotehost] --port [port]
```

Windows システム・サービスとしての Sybase ETL サーバの起動

Sybase ETL サーバを Windows システム・サービスとしてインストールして実行できます。Windows GUI とは無関係にシステム・サービスとして Sybase ETL サーバを実行するには、SYSTEM ユーザ・アカウントを使用してシステムを起動した後に ETL サーバを起動します。

注意 システム・サービスのインストール、削除、起動、停止を行うには、管理者権限が必要です。

Windows システム・サービスとしてサーバをインストールするには、コマンド・プロンプトで次のように入力します。

```
GridNode.exe --install [additional parameters]
```

サービスを削除するには、コマンド・プロンプトで次のように入力します。

```
GridNode.exe --remove
```

Windows サービスとして ETL サーバを実行すると、基本的なイベント (エラーや成功メッセージなど) が Windows イベント・ログに書き込まれます。

コマンド・ライン・パラメータ

この項では、Sybase ETL サーバのすべてのコマンド・ライン・パラメータについて説明します。

使用可能なパラメータの概要を表示するには、コマンド・プロンプトで `GridNode --help` または `GridNode -h` と入力します。各パラメータの長い形式と短い形式がコンソールに表示されます。例は次のとおりです。

```
--version,  -V    Displays version information
```

注意 完全なパラメータ名の前には必ずマイナス記号が 2 つ付きます。省略形の前には 1 つしか付きません。

表 6-1：コマンド・ライン・パラメータ

コマンド	省略形	UNIX	Windows	説明
install	inst	yes	yes	アプリケーションを Unix デーモンまたは Windows サービスとしてインストールする。
remove	rm	yes	yes	デーモンまたはシステム・サービスの開始を削除する。
setoptions	so	no	yes	Windows サービスとして実行するときに使用するコマンド・ライン・オプションを設定する。
getoptions	go	no	yes	Windows サービスとして実行するときに使用するコマンド・ライン・オプションを表示する。
background	bg	yes	no	システム リソースを使用しすぎないバックグラウンド・プロセスを設定する。
no_pidfile	nopid	yes	no	デーモンのプロセス ID がサーバによって記録されるファイルを設定する。
console	con	yes	yes	コンソールに詳細なエラー情報とトレース・メッセージを表示する。
diagnosis	diag	yes	yes	アプリケーション環境をリストする。
tracelevel	tl	yes	yes	0 (トレースなし) ～ 5 (非常に詳細) のデバッグ・トレース・レベルを設定する。
server	s	yes	yes	使用するリモート・サーバを示す。
port	p	yes	yes	操作対象のポート番号を示す。
version	V	yes	yes	アプリケーションのバージョン情報を確認する。
help	h	yes	yes	ヘルプ情報を表示する。
licenses	ll	yes	yes	使用可能なライセンスとそのステータスに関する簡単な情報を確認する。
nodelist	nl	yes	yes	既知のピア・ノードをすべてリストする。
shutdown	sh	yes	yes	ノードを停止する。
プロジェクトおよびジョブを実行するためのコマンド・ライン・パラメータ				
dbhost *host*		yes	yes	リポジトリ・データベースのホスト名またはデータ・ソース名 (DSN)。
dbinterface *interface*		yes	yes	リポジトリ・データベース・インタフェース。

コマンド	省略形	UNIX	Windows	説明
dbdatabase *database*		yes	yes	リポジトリ・データベース名。
dbschema *schema*		yes	yes	リポジトリ・データベース・スキーマ。
dbuser *user*		yes	yes	リポジトリ・データベース・ユーザ。
dbpassword *encrypted password*		yes	yes	リポジトリ・データベース・パスワード。
client *client*		yes	yes	リポジトリ・クライアント名。
user *user*		yes	yes	リポジトリ・クライアント・ユーザ。
password *encrypted password*		yes	yes	リポジトリ・クライアント・パスワード。
project [*name* | *ID*]		yes	yes	プロジェクト名または ID を指定してプロジェクトを実行する。
job [*name* | *ID*]		yes	yes	プロジェクト名または ID を指定してジョブを実行する。
paramset [*name* | *ID*]		yes	yes	プロジェクトまたはジョブの名前または ID によってパラメータ・セットを指定する。
encrypt *password*		yes	yes	パスワードを暗号化し、暗号化された値を表示する。dbpassword および password と共に使用する暗号化されたパスワードを生成するには、encrypt を使用する。
ping *host:port*		yes	yes	指定したホストとポートで ETL サーバが実行されているかどうかを確認する。
env "*variable1=value*; *variable2=value*; ...;*variableN=value*"		yes	yes	ETL サーバで実行する追加の環境変数を指定する。セミコロンを使用して複数の環境変数を区切り、変数の文字列全体を二重引用符で囲む。

ETL サーバを使用したプロジェクトおよびジョブの実行

Sybase ETL サーバは、表 6-1 (228 ページ) のコマンド・ライン・パラメータを使用して、サポートされているすべてのプラットフォームでプロジェクトおよびジョブを実行できます。プロジェクトおよびジョブを実行するには、次の構文を使用します。

```
GridNode --project PROJ-1234-5678 --dbinterface dbodbc --dbhost etl_comp
--client transformer--user TRANSFORMER --password 1234ABCD
```

ここで、プロジェクト ID は “PROJ-1234-5678”、データベース・インタフェースは “dbodbc”、ホスト名は “etl_comp”、クライアントは “transformer”、ユーザは “TRANSFORMER”、パスワードの暗号化バージョンは “1234ABCD” です。

また、次のコマンド・ライン・パラメータを使用してプロジェクトおよびジョブを実行することもできます。

- project − 名前によってプロジェクトおよびパラメータ・セットを指定します。名前によってジョブを指定することもできます。名前を指定する方が、複雑なプロジェクト、ジョブ、またはパラメータ・セット ID を入力するより簡単です。この例では、プロジェクト名 “LoadCustomers” とパラメータ・セット名 “myparams” を使用します。

```
GridNode --project LoadCustomers --dbinterface dbodbc --dbhost etl_com
--client transformer --user TRANSFORMER --paramset myparams --password 1234ABCD
```

- encrypt − 暗号化されたパスワードを生成します。暗号化されたパスワードは、dbpassword および password と共に使用します。“mypassword” を暗号化するには、次のように入力します。

  ```
  Gridnode --encrypt mypassword
  ```

  ETL サーバによって、暗号化されたパスワードが生成および表示されます。

- ping − ETL サーバが特定のホストおよびポートで実行されているかどうかを確認します。ETL サーバが “localhost” のデフォルト・ポートで実行されているかどうかを確認するには、次のように入力します。

  ```
  Gridnode --ping localhost
  ```

  ETL サーバが動作している場合は、次のメッセージが表示されます。

  ```
  localhost is alive!
  ```

  指定したホストおよびポートで ETL サーバが実行されていない場合は、エラー・メッセージが表示されます。

- env − プロジェクトおよびジョブの環境変数を指定します。uGetEnv 関数を使用して、実行時にこれらの変数の値にアクセスできます。この例では、“LoadCustomers” プロジェクトを環境変数 INPUT_FILE および OUTPUT_FILE と共に使用します。

```
GridNode --project LoadCustomers --dbinterface dbodbc --dbhost etl_com
--client transformer --user TRANSFORMER --paramset myparams --password
1234ABCD --env 'INPUT_FILE=input.txt;
OUTPUT_FILE=output.txt'
```

注意 コマンド・ラインを入力するときは、コマンド、パラメータ、および値を 1 行に続けて入力します。この項の例は、わかりやすくするために数行にまたがっています。

Sybase ETL バージョン 4.5 より前は、ProcessQ アプリケーションを使用してプロジェクトとジョブを実行していました。このアプリケーションは ETL サーバと共に配布され、Windows プラットフォームのみで内部的に使用されます。ProcessQ は現在廃止されており、下位互換性を保つためにのみ提供されています。ProcessQ の多くのパラメータは無効になり、使用できません。

注意 プロジェクトおよびジョブの実行に ProcessQ を使用しないことをおすすめします。

INI ファイルの設定

Sybase ETL サーバの設定が含まれている *INI* ファイルは、インストール・ディレクトリの *etc* フォルダにあります。

Default.ini

グループ	キー	値	デフォルト	説明
Network	proxy	host:port explorer	explorer	インターネット・アクセス用のプロキシを設定する。 “http_proxy”、“https_proxy”、“ftp_proxy” または “ftps_proxy” のキーを使用して、特定のプロトコル (HTTP、HTTPS、FTP、FTPS) のプロキシを微調整できる。 プロキシ値 “explorer” は、Windows 環境のシステム・プロキシを取得する。
Network	timeout	1-2147483 秒	600 秒	FTP 接続のタイムアウト値を設定する。

グループ	キー	値	デフォルト	説明
Language	Default	English_USA	English_USA	選択されている言語と国に基づいて、アプリケーションの動作を調整する。
Logging	Console	1/0	0	コンソールにログ情報を送信する。
Logging	LogFile	1/0	1	*system.log* ファイルにログ情報を送信する。
Logging	Tracelevel	0-5	0	さまざまなレベルのデバッグ情報を表示する。レベル 0 では最小限の情報が、レベル 5 では最も詳しい情報が表示される。通常実行時には、この値を 0 に設定して、パフォーマンスへの影響を最小限に抑えてください。
Logging	Flushtime	1-n	1	内部ログ・フラッシュ間隔を秒数で指定する。

Web ブラウザを使用したプロジェクトとジョブのモニタリング

Web ブラウザを使用して、グリッド・アーキテクチャの一部である ETL Development グリッド・ノードを含むすべてのグリッド・ノードの状態をモニタリングできます。ETL Development から起動するプロジェクトとジョブをモニタリングする他に、次のことができます。

- リモート・ジョブの状態をモニタリングする。
- リモート・ジョブとリモート・プロジェクトをサスペンドしたりレジュームしたりする。
- ETL サーバのリモート・ログ・ファイルを表示する。

モニタリングを開始する前に、次の手順を実行します。

- ETL サーバが稼働していない場合は起動します。
- お使いのマシンに Internet Explorer (IE) 6.0 以降がインストールされていることを確認します。

❖ **プロジェクトとジョブのモニタリング**

1 Web ブラウザを開きます。

2 次のように入力します。

```
http://<hostname>:<port_number>
```

ここで、<*hostname*> は ETL サーバが稼働しているマシンのネットワーク名であり、<*port_number*> はノードが起動するポートです。デフォルトのポート番号は 5124 です。

モニタリング・ページが表示され、次のタブが表示されます。

- ノードの概要 − サーバが稼働しているマシンのホスト名とオペレーティング・システム、サーバで稼働中のジョブの数、サーバに割り当てられている CPU、メモリ、ディスク領域の各容量、PID、アカウント情報、製品名、バージョン番号など、現在稼働しているサーバに関する詳細情報を表示します。このタブには、アクティブな最近のジョブのリストもあります。
- アクティブなジョブ − 稼働中のジョブに関する詳細リストが表示されます。
- ジョブ履歴 − 前日以降に実行されたすべてのジョブのリストが示されます。
- ログ履歴 − システム・ログ履歴が示されます。
- ノードの概要 − 稼働中のすべてのサーバのリストが示されます。

アクティブなジョブの表示

[Active Jobs] タブをクリックします。名前、ステータス、ジョブ内のプロジェクト数、開始時刻、終了時刻、処理されたレコードの数など、稼働中のすべてのジョブがそれらの詳細と共に表示されます。

アクティブなジョブのサスペンド

[Active Jobs] タブまたは [Node Summary] タブで、サスペンドするジョブの横にある [Suspend] アイコンをクリックします。ジョブのステータスが [Suspended] に変わります。ジョブ内のすべてのプロジェクトもサスペンドされます。

ジョブのレジューム

1　[Active Jobs] タブまたは [Node Summary] タブで、履歴を表示するジョブを選択します。または、[Job History] タブをクリックします。

2　サスペンドしたジョブの横にある [Resume] アイコンをクリックします。

ジョブのステータスが [Running] に変わり、ジョブ内のすべてのプロジェクトもレジュームされます。

アクティブなジョブのキャンセル

[Summary] タブで、キャンセルするジョブの横にある [Cancel] アイコンをクリックします。そのジョブがリストから削除されます。

アクティブなプロジェクトのサスペンド

[Active Jobs] タブで、サスペンドするプロジェクトの横にある [Suspend] アイコンをクリックします。プロジェクトのステータスが [Suspended] に変わります。

プロジェクトのレジューム

1 [Active Jobs] タブで、サスペンドしたプロジェクトがあるジョブを選択します。または、[Job History] タブをクリックします。

2 サスペンドしたプロジェクトの横にある [Resume] アイコンをクリックします。プロジェクトのステータスが [Running] に変わり、プロジェクトがレジュームされます。

プロジェクトのキャンセル

1 [Active Jobs] タブで、アクティブなジョブのリストからジョブを選択します。

2 キャンセルするプロジェクトの横にある [Cancel] アイコンをクリックします。

プロジェクトのステータスが [Cancelled] に変わり、プロジェクトが停止します。

稼働中のすべてのサーバを表示する

[Node Overview] タブを選択します。稼働中のすべてのサーバのリストが、ステータス、ジョブ数、サーバに割り当てられている CPU、メモリ、ディスク領域の各容量などの詳細と共に表示されます。

サーバの停止

[Node Overview] タブで、サーバの横にある [Shutdown] アイコンをクリックします。そのサーバがリストから削除されます。

ログ履歴の表示

[Log History] タブを選択します。テーブルが表示され、タイムスタンプ、エラーのタイプ、エラー・メッセージなど、システム・ログの詳細がそこに表示されます。

必要に応じて、[Download] をクリックしてログ・ファイルをお使いのマシンにダウンロードします。

Sybase ETL サーバのトラブルシューティング

Sybase 製品の保守契約を結んでいるサポート・センタに問い合わせる前に、次の手順を実行します。

1　エラー・テキストを確認します。

2　ログ・ファイルを確認します。

3　システム・トレースを有効にして再度実行します。

4　バージョンおよびリビジョン番号とマシン ID を確認します。

構文：

GridNode --version

出力：

```
GridNode 4.8.0.27424
```

5　有効なライセンスを確認します。

構文：

GridNode --licenses

出力：

```
GridNode (4.8.0.27424)
Grid Node
--------------------------------------------------
Product ID: SybaseETLServer
Machine ID: 9TuA+igB6298Hys=
SYSAM ID  : 001111eb57f9 DISK_SERIAL_NUM=189e22a0
--------------------------------------------------
Install Date: Tuesday Feb 03 13:53:47 2009
--------------------------------------------------

File: sybase_etl_server.license
Product: Sybase ETL Server (SybaseETLServer)
Version: 4.8
License: ETL Components 4.8(ETL_SERVER)
Status: Valid
```

付 録　A

関数リファレンス

この付録では、Sybase ETL 関数のリファレンスを提供します。

集合

関数	説明
uAvg	すべての入力値の平均値を返す。
uMax	値のリストの最大値を返す。
uMin	値のリストの最小値を返す。

uAvg

説明 すべての入力値の平均値を返します。

構文 real uAvg(value, ...)

パラメータ *number value*
任意のデータ型の式または値のリスト

例
```
uAvg(1,2,3,4,5) // returns 3
```

uMax

説明 値のリストの最大値を返します。

構文 uMax(value,...)

パラメータ *number value*
任意のデータ型の式または値のリスト

例
```
uMax(1, 6, 4, -6) // returns 6

uMax("b", "A", "a") // returns "b"

uMax("2004-05_02", "2006-12-12", "1999-05-30") //
returns "2006-12-12"
```

uMin

説明 値のリストの最小値を返します。

構文 uMin(value, ...|)

パラメータ *number value*
任意のデータ型の式または値のリスト

例
```
uMin(1, 6, 4, -6) // returns -6

uMin("b", "A", "a") // returns "A"

uMin("2004-05-02", "2006-12-12", "1999-05-30")
//returns "1999-05-30"
```

ビット操作

関数	説明
uBitAnd	ビット処理 AND 演算
uBitOr	ビット処理 OR 演算
uBitXOr	排他的ビット処理 OR 演算
uBitNot	ビット処理 NOT 演算

uBitAnd

説明　ビット処理 AND 演算

構文　number uBitAnd(value, ...)

パラメータ　*number value*
任意のデータ型の式または値のリスト

例
```
uBitAnd(10, 3) // returns "2"
```

uBitOr

説明　ビット処理 OR 演算

構文　number uBitOr(value, ...)

パラメータ　*number value*
任意のデータ型の式または値のリスト

例
```
uBitOr(10, 3) // returns "11"
```

uBitXOr

説明 排他的ビット処理 OR 演算

構文 number uBitXOr(value1 , value2)

パラメータ *number value1, value2*
計算する値

例
```
uBitXOr(10, 3) // returns "9"
```

uBitNot

説明 ビット処理 NOT 演算

構文 number uBitNot(value)

パラメータ *number value*
数値の引数

例
```
uBitNot(10) // returns "-11"
```

ブール値

関数	説明
uIsAscending	各パラメータの値が以前のパラメータの値以上の場合に 1 を返す。
uIsBoolean	パラメータが 1、true、または yes の場合に 1 を返す。
uIsDate	パラメータを日付として解釈できる場合に 1 を返す。
uIsDescending	各パラメータの値が以前のパラメータの値以下の場合に 1 を返す。
uIsEmpty	パラメータが空または null の場合に 1 を返す。
uIsInteger	パラメータを整数値として解釈できる場合に 1 を返す。
uIsFloat	パラメータを浮動小数点値として解釈できる場合に 1 を返す。
uIsNull	パラメータが null の場合に 1 を返す。
uIsNumber	パラメータを数値として解釈できる場合に 1 を返す。
uNot	1 が入力された場合は 0 を返し、0 が入力された場合は 1 を返す。

uIsAscending

説明　各パラメータの値が以前のパラメータの値以上の場合に 1 を返します。

構文　number uIsAscending(params, ...)

パラメータ　*params*
任意のデータ型の式または値のリスト

例　複数の値が昇順かどうかを確認します。

```
uIsAscending("A", "B", "C") // returns 1

uIsAscending("A", "A", "C") // returns 1

uIsAscending("A", "C", "B") // returns 0


uIsAscending("1", "2", "3") // returns 1

uIsAscending("3", "2", "2") // returns 0

uIsAscending("2004-03-03", "2004-03-05", "2004-03-07")
// returns 1

uIsAscending("2004-03-03", "2004-03-07", "2004-03-05")
//returns 0
```

uIsBoolean

説明　パラメータが次のいずれかの場合に 1 を返します。

- 1、true、または yes のいずれか
- 0、no、または false のいずれか

構文　number uIsBoolean(param)

パラメータ　*param*
任意のデータ型の式または値

例

ブール値を確認します。

```
uIsBoolean("1")     // returns 1
uIsBoolean("yes")   // returns 1
uIsBoolean("true")  // returns 1
uIsBoolean("-1")   // returns 0
uIsBoolean("0")  // returns 1
```

uIsDate

説明

パラメータを日付として解釈できる場合に 1 を返します。2 番目のパラメータを省略した場合は、次のフォーマットのいずれかが適用されます。

- `y-M-D H:N:S.s`
- `y-M-D H:N:S`
- `y-M-D`
- `H:N:S`

注意 フォーマット文字列の詳細については、uConvertDate 関数を参照してください。

構文

number uIsDate(datestring [, format])

パラメータ

string datestring
確認する文字列

string format (オプション)
入力日のフォーマット

例

```
uIsDate("2004-02-29") // returns 1
uIsDate("2003-02-29") // returns 0, since 2003 was not
a leap year
```

uIsDescending

説明　各パラメータの値が以前のパラメータの値以下の場合に 1 を返します。

構文　number uIsDescending(params, ...)

パラメータ　*params*

任意のデータ型の式または値のリスト

例　複数の値が降順かどうかを確認します。

```
uIsDescending("C", "B", "A") // returns 1

uIsDescending("C", "C", "A") // returns 1

uIsDescending("A", "C", "B") // returns 0

uIsDescending("3", "2", "1") // returns 1

uIsDescending("3", "2", "3") // returns 0

uIsDescending("2004-03-20", "2004-03-15", "2004-03-
07") // returns 1

uIsDescending("2004-03-20", "2004-03-07", "2004-03-
15") // returns 0
```

uIsEmpty

説明　パラメータが空または null の場合に 1 を返します。

構文　number uIsEmpty(param)

パラメータ　*param*

調査する式または値

例

```
uIsEmpty("1")           // returns 0

uIsEmpty(null)          // returns 1

uIsEmpty("")            // returns 1
```

uIsInteger

説明 パラメータを整数値として解釈できる場合に 1 を返します。

構文 number uIsInteger(param)

パラメータ *param*

調査する式または値

例

```
uIsInteger ("1")    // returns 1
uIsInteger ("2.34") // returns 0
uIsInteger ("ABC")  // returns 0
```

uIsFloat

説明 パラメータを浮動小数点値として解釈できる場合に 1 を返します。

構文 number uIsFloat(param)

パラメータ *param*

調査する式または値

例

```
uIsFloat("1")    // returns 1
uIsFloat("2.34") // returns 1
uIsFloat("ABC")  // returns 0
```

uIsNull

説明 パラメータが null の場合に 1 を返します。

構文 number uIsNull(param)

パラメータ *param*

調査する式または値

例

```
uIsNull("1")  // returns 0
uIsNull(null) // returns 1
```

uIsNumber

説明　パラメータを数値として解釈できる場合に 1 を返します。

構文　number uIsNumber(param)

パラメータ　*param*
　調査する式または値

例　数値を確認します。

```
uIsNumber("1")     // returns 1
uIsNumber("2.34") // returns 1
uIsNumber("ABC")   // returns 0
```

uNot

説明　1 が入力された場合は 0 を返し、0 が入力された場合は 1 を返します。この関数は、uIs- 関数と必ず併用します。返されるブール値は true や false ではなく、0 や 1 になるためです。

構文　number uNot(expression)

パラメータ　*expression*
　負にする数値

例

```
uNot(1) // returns 0
```

変換

関数	説明
uBase64Decode	文字列を Base64 表現から復号する。
uBase64Encode	文字列を Base64 表現にコード化する。
uConvertDate	日付文字列をデフォルトまたはカスタムの日付フォーマットに変換する。
uFromHex	16 進数値を整数値に変換する。
uToHex	整数値を 16 進数に変換する。
uHexDecode	文字列を 16 進数値から構成する。
uHexEncode	文字列の文字を 16 進数表記にコード化する。
uToUnicode	文字列を Unicode 表現に変換する。

関数	説明
uURIDecode	文字列を復号し、エスケープ・シーケンスを元の値に置換する。
uURIEncode	URI の特定の文字をエスケープ・シーケンスに置換する。

uBase64Decode

説明 文字列を Base64 表現から復号します。

構文 string uBase64Decode(input)

パラメータ *string input*
復号する文字列

例
```
uBase64Decode("QSBzZWNyZXQ=") // returns "A secret"
```

uBase64Encode

説明 文字列を Base64 表現にコード化します。

構文 string uBase64Encode(input)

パラメータ *string input*
コード化する文字列

例
```
uBase64Encode("A secret") // returns "QSBzZWNyZXQ="
```

uConvertDate

説明 日付文字列をデフォルトまたはカスタムの日付フォーマットに変換します。最初のパラメータは、変換する日付文字列で、2 番目のパラメータは、入力日付文字列の日付フォーマットを指定するフォーマット文字列です (次の表を参照)。outputformat パラメータはオプションで、省略した場合、日付は `y-M-D H:N:S` のフォーマットに変換されます。

この関数は、*1582* 年から現在の日までの日付を処理します。日付を変換できない場合、結果の文字列は空になります。

構文

string uConvertDate(datestring, inputformat [, outputformat])

パラメータ

string datestring
　変換する日付文字列

string inputformat
　入力文字列の日付／時刻フォーマット

string outputformat (オプション)
　目的の出力フォーマット。省略した場合のデフォルトのフォーマットは `y-M-D H:N:S` です。

例

日付文字列を別のフォーマットに変換します。

```
uConvertDate("2005-06-27 00:00:00","y-M-D H:N:S","D mY") // returns "27 JUN 05"

uConvertDate("27 JUN 05", "D m Y") // returns "2005-06-27 00:00:00"
```

使用法

説明

uConvertDate 関数は、ソース・フォーマットと送信先フォーマット文字列を使用して、日付文字列を別のフォーマットに変換します。最初のパラメータは、変換する日付文字列です。2 番目のパラメータは、入力日の日付フォーマットを指定するフォーマット文字列です (以下のリストを参照してください)。outputformat パラメータはオプションです。省略した場合、日付は `y-M-D H:N:S` のフォーマットを使用して変換されます。この関数は、*1582* 年から現在の日までの日付を処理します。日付を変換できなかった場合、結果の文字列は空になります。

識別子の説明

識別子	説明
`Y`	2 桁の年 (06)
`y`	4 桁の年 (2006)
`C`	世紀 (20)
`M`	月 (03)
`m`	月 (JUN)
`D`	日 (12)
`H`	時 (00 ～ 23)
`h`	時 (01 ～ 12)
`N`	分
`S`	秒
`s`	百分の 1 秒
`t`	千分の 1 秒
`A`	AM/PM
`d`	曜日 (5)

識別子	説明
D	曜日 (Friday)
E	年間通算日 (001 ～ 366)
G	年間通算週 (01 ～ 52)
F	月間通算週 (1 ～ 6)

uFromHex

説明 16 進数値を整数値に変換します。

構文 integer uFromHex(input)

パラメータ *string input*

変換する文字列

例

```
uFromHex("A3F") // returns 2623
uFromHex("B")   // returns 11
```

uToHex

説明 整数値を 16 進数に変換します。

構文 string uToHex(input)

パラメータ *integer input*

変換する整数値

例

```
uToHex(45) // returns "2D"
```

uHexDecode

説明 文字列を 16 進数値から構成します。

構文 string uHexDecode(input)

パラメータ *string input*

16 進数値を含む 16 進文字列

例　16 進数値を文字列に変換します。

```
uHexDecode("313730") // returns "170"
uHexDecode(313730)   // returns "170"
```

uHexEncode

説明　文字列の文字を 16 進数表記にコード化します。

構文　string uHexEncode(input)

パラメータ　*string input*
　コード化する文字列

例　文字列を 16 進数値に変換します。

```
uHexEncode("170") // returns "313730"
uHexEncode(170)   // returns "313730"
```

uToUnicode

説明　文字列を Unicode 表現に変換します。

構文　string uToUnicode(input)

パラメータ　*string input*
　入力文字列

uURIDecode

説明　文字列を復号し、エスケープ・シーケンスを元の値に置換します。

構文　string uURIDecode(uri)

パラメータ　*string uri*
　復号する URI

例

```
uURIDecode("www.myServer.com/filename%20with%20spaces.
txt") // returns
"www.myServer.com/filename with spaces.txt"
```

uURIEncode

説明 URI の特定の文字をエスケープ・シーケンスに置換します。

構文 string uURIEncode(uri)

パラメータ *string uri*

コード化する URI

例

```
uURIEncode("www.myServer.com/filename with
spaces.txt")// returns
"www.myServer.com/filename%20with%20spaces.txt"
```

日付と時刻

ほとんどの**日付**と**時刻**の関数は、uFormatDate 関数から派生したものです。唯一の違いは、他の**日付**と**時刻**の関数は、特殊なフォーマットまたは一部の日付のみを返し、最初のフォーマット・パラメータを持たないことです。したがって、uDate() は uFormatDate("%Y-%m-%d") と同等です。

参照

- 「時刻文字列」(250 ページ)
- 「変更子」(251 ページ)
- 「日付と時刻の計算」(252 ページ)
- 「既知の制限」(253 ページ)
- 「日付と時刻の関数リスト」(253 ページ)

時刻文字列

説明 時刻文字列には、次のいずれかのフォーマットを使用できます。

1 *YYYY-MM-DD*
2 *YYYY-MM-DD HH:MM*
3 *YYYY-MM-DD HH:MM:SS*
4 *YYYY-MM-DD HH:MM:SS.SSS*
5 *HH:MM*
6 *HH:MM:SS*
7 *HH:MM:SS.SSS*
8 now
9 *DDDD.DDDD*

注意
時刻のみを指定するフォーマット 5 ～ 7 は、2000-01-01 の日付を想定します。フォーマット 8 は、万国標準時 (UTC) を使用して現在の日付と時刻に変換されます。フォーマット 9 は浮動小数点値として表現されるユリウス日数です。

例

現在の時刻を取得します。 日付が指定されない場合は、時刻文字列 now が想定され、日付は現在の日時に設定されます。

```
uDate() // returns something like "2006-03-01"
uDate() is equivalent to uDate("now")
```

特殊日の取得

```
uDate("2004-01-04 14:26:33")
// returns the date part "2004-01-04"
```

変更子

説明

時刻文字列の後には、日付または日付の解釈を変更する 0 または変更子を指定できます。使用可能な変更子は以下のとおりです。

1 *NNN* days
2 *NNN* hours
3 *NNN* minutes
4 *NNN.NNNN* seconds
5 *NNN* months
6 *NNN* years
7 start of month
8 start of year
9 start of day
10 weekday *N*
11 unixepoch
12 localtime
13 utc

例

最初のサイズ変更子 (1 ～ 6) は、指定された時間量を、前の時刻文字列で指定された日付に追加します。

“start of” 変更子 (7 ～ 9) は、日付を現在の月、年、または日の開始に戻します。

“weekday” (10) 変更子は、曜日番号が *N* である次の日まで日付を先送りします。日曜日は 0、月曜日は 1 になり、以降同様に続きます。

unixepoch 変更子 (11) は、それが *DDDD.DDDDD* フォーマットの時刻文字列の直後に指定された場合にのみ機能します。この変更子は、*DDDD.DDDDD* が通常どおりにユリウス日数値として解釈されるのではなく、1970 年から始まる秒数値として解釈されるようにします。この変更子を使用すると、UNIX ベースの時刻をユリウス日数に簡単に変換できます。

localtime 変更子 (12) は前の時刻文字列を調整し、正しいローカル時刻が表示されるようにします。utc はこれを元に戻します。

日付と時刻の計算

説明

次の例は、一般的な日付と時刻の計算を示します。

例

現在の日付を計算します。

```
uDate('now')
```

現在の月の最後の日を計算します。

```
uDate('now','start of month','+1 month','-1 day')
```

UNIX のタイムスタンプ 1092941466 の日付と時刻を計算します。

```
uDatetime(1092941466, 'unixepoch')
```

UNIX のタイムスタンプ 1092941466 の日付と時刻を計算し、ローカル・タイム・ゾーンに合わせて調整します。

```
uDatetime(1092941466, 'unixepoch', 'localtime')
```

現在の UNIX のタイムスタンプを計算します。

```
uFormatDate ('%s','now')
```

2 つの日付間の秒数を計算します。

```
uJuliandate('now')*86400 - uJuliandate ('2004-01-01 02:34:56')*86400
```

現在の年の 10 月 (1 月 + 9) の第 1 火曜日の日付を計算します。

```
uDate('now','start of year','+9 months','weekday 2')
```

既知の制限

説明

ローカル時刻の計算は、ロケールによって異なります。この実装では、標準の C ライブラリ関数 localtime() を使用してローカル時刻の計算を行います。また、localtime() C 関数は通常 1970 ～ 2037 年の範囲でのみ機能します。この範囲外の日付の場合は、年をこの範囲内の対応年にマッピングし、計算を行ってから再度マッピングして正しい年に戻します。

- ユリウス日数 0 (-4713-11-24 12:00:00) より前の日付を計算した場合は正しい結果が生まれません。
- すべての内部計算ではグレゴリオ暦システムが想定されます。

日付と時刻の関数リスト

説明

次の表は、日付と時刻の関数をすべて示します。

関数	説明
uDate	日付の年、月、日を *YYYY-MM-DD* のフォーマットで返す。
uDateTime	日付の年、月、日を *YYYY-MM-DD HH.MM.SS* のフォーマットで返す。
uDay	指定されている日付の日数を返す。
uDayOfYear	年の開始から数えた日数を返す。
uHour	指定されている日付の時間を返す。
uQuarter	四半期を返す。
uIsoWeek	ISO 8601 によって定義されている週番号を返す。
uJuliandate	グリニッジにおける紀元前 4714 年 11 月 24 日正午から数えた日数を *DDDD.DDDD* のフォーマットで返す。
uMinute	指定されている日付の分を返す。
uMonth	指定されている日付の月を返す。
uMonthName	現在のロケール言語で指定されている日付の月名を返す。
uMonthNameShort	現在のロケール言語で指定されている日付の月名の省略形を返す。
uSeconds	指定されている日付の秒を返す。
uTime	日付の時刻の部分を *HH.MM.SS* のフォーマットで返す。
uTimeDiffMs	2 つの日付の差をミリ秒単位で返す。

関数	説明
uWeek	指定されている日付の週を返す。
uWeekday	指定されている日付の曜日を返す。
uWeekdayName	現在のロケール言語で指定されている日付の曜日名を返す。
uWeekdayNameShort	現在のロケール言語で指定されている日付の曜日名の省略形を返す。
uYear	指定されている日付の年を返す。

注意 使用可能な変更子の引数の詳細については、「日付と時刻」(250 ページ) を参照してください。

uDate

説明 日付の年、月、日を *YYYY-MM-DD* のフォーマットで返します。

構文 string uDate([modifiers])

パラメータ *string modifiers* (オプション)
日付または日付計算を指定した文字列のリスト。デフォルトは now 変更子です。

例 タイムスタンプの日付部分を取得します。

```
uDate("now") // returns current date in the form "YYYY-MM-DD".

uDate("now", "start of year", "9 months", "weekday 2") // returns the date of the first Tuesday in October this year.
```

uDateTime

説明 日付の年、月、日を *YYYY-MM-DD HH.MM.SS* のフォーマットで返します。

構文 string uDateTime([modifiers])

パラメータ *string modifiers* (オプション)
日付または日付計算を指定した文字列のリスト。デフォルトは now 変更子です。

例　タイムスタンプの日付時刻を取得します。

```
uDateTime("now") // returns current date in the form
"YYYY-MM-DD HH:MM:SS"

uDateTime("now", "start of month", "1 months", "-1 day")
// returns the date of the last day in this month
```

uDay

説明　指定されている日付の日数を返します。

構文　string uDay([modifiers])

パラメータ　*string modifiers* (オプション)
日付または日付計算を指定した文字列のリスト。デフォルトは now 変更子です。

例　タイムスタンプから日数を取得します。

```
uDay("now") // returns current day number

uDay("1969-03-13 10:22:23.231") // returns "13"
```

uDayOfYear

説明　年の開始から数えた日数を返します。

構文　string uDayOfYear([modifiers])

パラメータ　*string modifiers* (オプション)
日付または日付計算を指定した文字列のリスト。デフォルトは now 変更子です。

例　タイムスタンプから日数を取得します。

```
uDayOfYear("now") // returns how many days have already
passed this year

uDayOfYear("1969-03-13 10:22:23.231") // returns "72"
```

uHour

説明 指定されている日付の時間を返します。

構文 string uHour([modifiers])

パラメータ *string modifiers* (オプション)
日付または日付計算を指定した文字列のリスト。デフォルトは now 変更子です。

例

```
uHour("now") // returns current hour

uHour("1969-03-13 10:22:23.231") // returns "10"
```

uQuarter

説明 四半期を返します。

構文 string uQuarter([modifiers])

パラメータ *string modifiers* (オプション)
日付または日付計算を指定した文字列のリスト。デフォルトは now 変更子です。

例

```
uQuarter ("now") // returns current quarter

uQuarter ("2005-03-13 10:22:23.231") // returns "1"
```

uIsoWeek

説明 *ISO 8601* によって定義されている週番号を返します。
年の最初の週の番号は *01* で、これは暦年の最初の木曜日を含む週として定義されています。これはさらに、次のことを示します。

- ほとんどが暦年内にある最初の週
- 1 月 4 日を含む週
- 1 月 1 日に最も近い月曜日から始まる週

年の最後の週の番号は 52 または 53 であるので、以下のようになります。

- 暦年の最後の木曜日を含む週
- ほとんどが暦年内にある最後の週
- 12 月 28 日を含む週
- 12 月 31 日に最も近い日曜日で終わる週

構文　number uIsoWeek([modifiers])

パラメータ　*string modifiers* (オプション)
日付または日付計算を指定した文字列のリスト。デフォルトは now 変更子です。

例

```
uIsoWeek("now") // returns current week number
```

uJuliandate

説明　グリニッジにおける紀元前 4714 年 11 月 24 日正午から数えた日数を *DDDD.DDDD* のフォーマットで返します。日付と時刻の計算には、juliandate 関数が最適な選択肢です。

構文　string uJuliandate([modifiers])

パラメータ　*string modifiers* (オプション)
日付または日付計算を指定した文字列のリスト。デフォルトは "now" 変更子です。

例　日付を計算のために数値に変換します。

```
uJuliandate("now") // returns current date in the form
"DDDD.DDDD"
```

2 つの日付間の秒数を計算します。

```
uJuliandate('now')*86400 - uJuliandate ('2004-01-01
02:34:56')*86400
```

ヘースティングズの戦いが行われた日から経過した日数を計算します。

```
uJuliandate('now') - uJuliandate('1066-10-
14','gregorian')
```

UNIX のタイムスタンプ 1092941466 の日付と時刻を計算し、ローカル・タイム・ゾーンに合わせて調整します。

```
uJuliandate(1092941466, 'unixepoch', 'localtime')
```

uMinute

説明　指定されている日付の分を返します。

構文　string uMinute([modifiers])

パラメータ *string modifiers* (オプション)
日付または日付計算を指定した文字列のリスト。デフォルトは now 変更子です。

例
```
uMinute("now") // returns current minute
uMinute("1969-03-13 10:22:23.231") //returns "22"
```

uMonth

説明 指定されている日付の月を返します。

構文 string uMonth([modifiers])

パラメータ *string modifiers* (オプション)
日付または日付計算を指定した文字列のリスト。デフォルトは now 変更子です。

例
```
uMonth("now") // returns current month
uMonth("1969-03-13 10:22:23.231") // returns "03"
```

uMonthName

説明 現在のロケール言語で指定されている日付の月名を返します。

構文 string uMonthName([modifiers])

パラメータ *string modifiers* (オプション)
日付または日付計算を指定した文字列のリスト。デフォルトは now 変更子です。

例 日付から月名を取得します。

```
uMonthName("now") // returns current name of month
```

ロケールを英語に設定します。

```
uSetLocale("English")
uMonthName("1969-03-13 10:22:23.231") // returns
"March"
```

ロケールをドイツ語に設定します。

```
uSetLocale("German")
uMonthName("1969-03-13 10:22:23.231") // returns "März"
```

uMonthNameShort

説明　現在のロケール言語で指定されている日付の月名の省略形を返します。

構文　string uMonthNameShort([modifiers])

パラメータ　*string modifiers (オプション)*
日付または日付計算を指定した文字列のリスト。デフォルトは now 変更子です。

例　日付から月名を取得します。

```
uMonthNameShort("now") // returns current name of
month.
```

ロケールを英語に設定します。

```
uSetLocale("English")
uMonthNameShort("1969-03-13 10:22:23.231") // returns
"Mar"
```

ロケールをドイツ語に設定します。

```
uSetLocale("German")
uMonthNameShort("1969-03-13 10:22:23.231") // returns
"Mär"
```

uSeconds

説明　指定されている日付の秒を返します。

構文　string uSeconds([modifiers])

パラメータ　*string modifiers* (オプション)
日付または日付計算を指定した文字列のリスト。デフォルトは now 変更子です。

例

```
uSeconds("now") // returns current second
```

```
uSeconds("1969-03-13 10:22:23.231") // returns "23"
```

uTime

説明　日付の時刻の部分を *HH.MM.SS* のフォーマットで返します。

構文　string uTime([modifiers])

パラメータ *string modifiers* (オプション)
日付または日付計算を指定した文字列のリスト。デフォルトは now 変更子です。

例 タイムスタンプの時刻部分を取得します。

```
uTime() // returns current UTC time

uTime("now","localtime") // returns current local time
```

uTimeDiffMs

説明 2 つの日付の差をミリ秒単位で返します。

構文 string uTimeDiffMs(date1, date2)

パラメータ *string date1*
古い方の日付
string date2
新しい方の日付

例

```
uTimeDiffMs ("18:34:20", "18:34:21")     // returns 1000

uTimeDiffMs ("18:34:20", "18:34:21.200") // returns
1200
```

uWeek

説明 指定されている日付の週を返します。

構文 string uWeek([modifiers])

パラメータ *string modifiers* (オプション)
日付または日付計算を指定した文字列のリスト。デフォルトは now 変更子です。

例

```
uWeek("now") // returns current week

uWeek("1969-03-13 10:22:23.231") // returns "10"
```

uWeekday

説明　指定されている日付の曜日番号を返します。

構文　string uWeekday([modifiers])

パラメータ　*string modifiers* (オプション)
日付または日付計算を指定した文字列のリスト。デフォルトは now 変更子です。

例

```
uWeekday("now") // returns current weekday number

uWeekday("1969-03-13 10:22:23.231") // returns "4" for
Thursday
```

uWeekdayName

説明　現在のロケール言語で指定されている日付の曜日名を返します。

構文　string uWeekdayName([modifiers]);

パラメータ　*string modifiers* (オプション)
日付または日付計算を指定した文字列のリスト。デフォルトは now 変更子です。

例

```
uWeekdayName("now") // returns current weekday name
```

ロケールを英語に設定します。

```
uSetLocale("English")
uWeekdayName("1969-03-13 10:22:23.231") // returns
"Thursday"
```

ロケールをドイツ語に設定します。

```
uSetLocale("German")
uWeekdayName("1969-03-13 10:22:23.231") // returns
"Donnerstag"
```

uWeekdayNameShort

説明 現在のロケール言語で指定されている日付の曜日名の省略形を返します。

構文 string uWeekdayNameShort([modifiers])

パラメータ *string modifiers (オプション)*
日付または日付計算を指定した文字列のリスト。デフォルトは now 変更子です。

例

```
uWeekdayNameShort("now") // returns current weekday
name
```

ロケールを英語に設定します。

```
uSetLocale("English")
uWeekdayNameShort("1969-03-13 10:22:23.231") //returns
"Thu"
```

ロケールをドイツ語に設定します。

```
uSetLocale("German")
uWeekdayNameShort("1969-03-13 10:22:23.231") //returns
"Don"
```

uYear

説明 指定されている日付の年を返します。

構文 string uYear([modifiers])

パラメータ *string modifiers* (オプション)
日付または日付計算を指定した文字列のリスト。デフォルトは now 変更子です。

例

```
uYear("now") // returns current year
```

```
uYear("1969-03-13 10:22:23.231") // returns "1969"
```

エラー処理

関数	説明
uError	エラー・テキストをログに書き込み、エラーを通知する。
uErrortext	最後のエラー・メッセージを返す。
uWarning	警告メッセージをログに書き込む。
uInfo	情報メッセージをログに書き込む。
uTrace	トレース・メッセージをログに書き込む。
uTracelevel	トレース・メッセージの詳細レベルをログに設定する。

uError

説明　エラー・テキストをログに書き込み、エラーを通知します。

構文　string uError(errortext)

パラメータ　*string errortext*

ログ・ファイルに書き込むテキスト

例　エラーを通知します。

```
uError("'PP' is no valid country key.")
```

uErrortext

説明　最後のエラー・メッセージを返します。

構文　string uErrortext()

例

```
uErrortext() // returns last error text
```

uInfo

説明　情報メッセージをログに書き込む。

構文　string uInfo(infotext)

パラメータ *string infotext*
ログ・ファイルに書き込むテキスト

例 情報メッセージのログを記録します。

```
uInfo("21445 records selected.")
```

uWarning

説明 警告メッセージをログに書き込みます。

構文 string uWarning(warningtext)

パラメータ *string warningtext*
ログ・ファイルに書き込むテキスト

例 警告メッセージのログを記録します。

```
uWarning("The attribute for the customer name is null.")
```

uTrace

説明 トレース・メッセージをログに書き込みます。

uTrace() 関数を呼び出す前に、トレース・レベルを手動で 1 以上に設定する必要があります。次のいずれかの方法でトレース・レベルを 1 に設定できます。

- uTrace() 関数を呼び出す前に uTracelevel(1) を呼び出す。
- ETL Development を使用している場合は、インストール・フォルダの *etc* ディレクトリにある *Default.ini* ファイルでトレース・レベルを 1 に設定します。ETL Development を再起動します。
- ETL サーバを使用している場合は、“--tracelevel 1” オプションを使用してサーバを起動します。

構文 string uTrace(tracetext);

パラメータ *string tracetext*
ログ・ファイルに書き込むテキスト

例

```
uTrace("CUSTOMER_NAME = " + CUSTOMER_NAME)
```

uTracelevel

説明　トレース・メッセージの詳細レベルをログに設定します。tracelevel の範囲は 0 (トレースなし) ～ 5 (非常に冗長) です。

構文　uTracelevel(tracelevel)

注意 ログを多く使用すると、実行速度が大幅に遅くなります。

パラメータ　*integer tracelevel*
トレース・メッセージの詳細度を指定します (0 = オフ、5 = 非常に冗長)。

例
```
uTracelevel(5) // sets the tracelevel to 'very verbose'
```

ファイル

関数	説明
uFileInfo	ファイルに関する情報を返す。
usFileRead	データをファイルから読み込む。
uFileWrite	データをファイルに書き込む。

uFileInfo

説明　ファイルに関する情報を返します。infotype が EXISTS に設定されているとき、ファイルが存在する場合はファイルのパス全体が返され、存在しない場合は空の文字列が返されます。infotype が SIZE に設定されているときは、ファイルのサイズが返されます。ファイルが存在しない場合は、空の文字列が返されます。

注意 JavaScript 環境ではバックスラッシュはエスケープ・シーケンスに使用されるため、バックスラッシュを 2 つ続けて指定する必要があることに注意してください。

構文　string uFileInfo(file [, infotype])

パラメータ　*string file*
調査するファイル

string infotype (オプション)
取得する情報の種類。デフォルトは EXISTS です。

例

ファイル情報を取得します。

```
uFileInfo("C:¥¥windows¥¥notepad.exe") // returns
C:¥windows¥notepad.exe

uFileInfo("C:¥¥windows¥¥notepad.exe","SIZE") // returns
68608
```

usFileRead

説明

データをファイルから読み込みます。

構文

string uFileRead(URL [, bytes] [, offset] [, encoding])

パラメータ

string URL
読み込むソースを指定した URL

integer bytes (オプション)
読み込むバイト数。デフォルトは、ファイル全体を意味する *0* です。

integer offset (オプション)
ファイルの先頭から省略するバイト数。デフォルトは *0* です。

string encoding (オプション)
データ ソースのコード化。デフォルトのコード化は *ISO8859-1* です。

例

ローカル・ファイルにアクセスします。

```
uFileRead("c:¥¥myFile.txt")

uFileRead("/home/testuser/myfile.txt")

uFileRead("c:¥¥myFile.txt")
```

ファイルを Windows 共有から読み込みます。

```
uFileRead("¥¥¥¥fileserver¥¥freeShare¥¥testfile.txt")
```

ファイルのコンテンツを HTTP と HTTPS を経由して読み込みます。

```
uFileRead("http://www.google.com/search?hl=en&qpizza&b
tnG=Google+Search")

uFileRead("https://www.google.com/search?hl=en&q=pizza
&btnG=Google+Search")
```

ファイルのコンテンツを FTP を経由して読み込みます。

```
uFileRead("ftp://myUser:myPasswd@myServer/data/myFile.
txt")
```

uFileWrite

説明　データをファイルに書き込みます。URL が指定されなかった場合、データは Sybase ETL ログ・ディレクトリ内のファイル *write.log* に書き込まれます。

構文　string uFileWrite(data [, URL] [, append] [, encoding])

パラメータ

string data
　書き込むデータ

string URL (オプション)
　ファイルのアクセスおよびロケーション用 URL

number append (オプション)
　データを追加するかどうかを示すフラグ (0/1)

string encoding (オプション)
　対象ファイルのコード化

例　データを CIFS 経由でファイルに書き込みます。

```
uFileWrite("hello",
"//myServer/myShare/data/test.txt")
```

フォーマット

関数	説明
uFormatDate	日付情報を含むユーザ定義文字列を返す。

uFormatDate

説明　日付情報を含むユーザ定義文字列を返します。参照先の日付部分に置換されるフォーマット文字列内の特殊エスケープ・シーケンスのリストについては、以下の「使用法」を参照してください。

構文　number uFormatDate(format, modifiers, ...)

パラメータ

string format
　戻り文字列のフォーマット指定

string modifiers (オプション)
　日付または日付計算を指定した文字列のリスト。デフォルトは now 変更子です。

例

文字列を日付から作成します。

```
uFormatDate("Today is %A the %d of %B in %Y", "now")
//returns something like "Today is Thursday the 10 of
February in 2005"
```

使用法

関数 uFormatDate は、日付情報を含むユーザ定義文字列を返します。フォーマット文字列内の特殊エスケープ・シーケンスは、参照先の日付部分に置換されます。

エスケープ・シーケンス

エスケープ・シーケンス	戻り値
%A	曜日名
%a	曜日名の短縮形
%B	月名
%b	月名の短縮形
%d	日
%f	小数秒 SS.SSS
%H	時間 00 ～ 24
%j	年間通算日 (000 ～ 366)
%J	ユリウス日数
%m	月
%M	分
%s	1970-01-01 から数えた秒
%S	秒 00 ～ 59
%w	曜日 0 ～ 6、0 は日曜日
%W	年間通算週
%Y	年 0000 ～ 9999
%%	%

ファジー検索

関数	説明
uGlob	ワイルドカードに UNIX ファイル展開構文を使用して、値の大文字と小文字の区別を比較する。
uLike	値の大文字と小文字の区別を比較する。
uMatches	指定された文字列が正規表現に一致した場合に true を返す。

uGlob

説明

ワイルドカードに UNIX ファイル展開構文を使用して、値の大文字と小文字の区別を比較します。

構文

bool uGlob(pattern, text)

パラメータ

string pattern
　一致パターンを記述した文字列

string text
　調査する文字列

例

UNIX ファイル展開構文を使用して値を比較します。

```
uGlob("Mr. *", "Mr. Smith")   // returns 1, indicating
a match

uGlob("Mr. *", "Mrs. Clarke") // returns 0
```

展開規則：

'*' − 0 文字以上のシーケンスと一致する

'?' − 1 文字と一致する

[^...] − カッコ内のリストにない 1 文字と一致する

[...] − カッコ内の文字のリストに含まれている 1 文字と一致する

[...] と [^...] の一致では、']' 文字を '[' または '^' の後の最初の文字にすることで、']' 文字をリストに含めることができます。文字の範囲は '-' を使用して指定できます。

例：

"[a-z]" は 1 つの小文字と一致します。'-' を一致させるには、これをリスト内の最後の文字にします。

'*' または '?' を一致させるには、これらを "[]" 内に配置します。

例：abc[*]xyz は "abc*xyz" のみと一致します。

uLike

説明

値の大文字と小文字の区別を比較します。

uLike 関数はパターン一致比較を行います。最初のパラメータはパターンを含んでおり、2 番目のパラメータはパターンに一致させる文字列を含んでいます。パターン内のパーセント記号 % は、文字列内の 0 以上の文字のシーケンスと一致します。パターン内のアンダースコア _ は、文字列内の任意の 1 文字と一致します。その他の文字は、それらと同じ文字または大文字と小文字が異なる同じ文字と一致します (たとえば大文字と小文字を区別しない一致など)。

注意 現時点では uLike は 7 ビットのラテン文字の大文字と小文字のみを理解するため、8 ビットの ISO8859 文字または UTF-8 文字では大文字と小文字が区別されます。たとえば、
`uLike('a' ,'A')` は 1
`uLike('æ' ,'Æ')` は 0

構文 number uLike(pattern, text)

パラメータ

string pattern
一致パターンを記述した文字列

string text
調査する文字列

例 パターン一致を使用して値を比較します。

```
uLike("% happy %", "A happy man.") // returns 1
uLike("% happy %", "A sad man.")   // returns 0
```

uMatches

説明 指定された文字列が正規表現に一致した場合に true を返します。

構文 number uMatches(text, regexpr)

パラメータ

string text
調査するテキスト

string regexpr
正規表現の指定

例 文字列を浮動小数点数値として解釈できるかどうかを確認します。

```
uMatches("abc","[-+]?[0-9]*¥¥.?[0-9]*")  // return 0
uMatches("1.23","[-+]?[0-9]*¥¥.?[0-9]*")  // return 1
```

参照

関数	説明
uChoice	インデックスで指定されたパラメータの値を返す。
uFirstDifferent	先頭のパラメータとは異なる最初のパラメータ値を返す。
uFirstNotNull	最初の null 以外のパラメータを返す。

関数	説明
uElements	デリミタ文字列内の要素の数を返す。
uToken	デリミタ文字列の N 番目の要素を返す。

uChoice

説明　インデックスで指定されたパラメータの値を返します。インデックス値は 0 から開始されるため、ゼロのインデックスは 2 番目のパラメータを返します。

構文　string uChoice(index, values, ...)

パラメータ

integer index

戻り値を参照しているインデックス番号。0 から開始。

string values

値のリスト

例

IF 構成体：

```
uChoice(0, "A", "B") // returns "A"
uChoice(1, "A", "B") // returns "B"
```

CASE 構成体：

```
uChoice(2, "n.a.", "Jan", "Feb", "Mar") //returns "Feb"
```

色の ID を対応する色の名前に置換する検索関数をシミュレートします。

```
uChoice(IN.Color, "n.a.", "Red", "Blue", "Green")
```

uFirstDifferent

説明　先頭のパラメータとは異なる最初のパラメータ値を返します。

構文　string uFirstDifferent(params, ...)

パラメータ

params

任意のデータ型の式または値のリスト

例

```
uFirstDifferent("2004-05-01", "2004-05-01", "2005-01-
04", "2005-11-24",) //returns "2005-01-04"
```

uFirstNotNull

説明 最初の null 以外のパラメータを返します。

構文 string uFirstNotNull(params, ...)

パラメータ *params*
任意のデータ型の式または値のリスト

例
```
uFirstNotNull(null, null , "A", "B") // returns "A"
```

uElements

説明 デリミタ文字列内の要素の数を返します。2 番目のパラメータを省略した場合は、スペース (ASCII 32) がデリミタとして使用されます。

構文 integer uElements(text [, delimiter])

パラメータ *string text*
調査する文字列

string delimiter (オプション)
使用するデリミタ。デフォルトのデリミタはスペース文字です。

例 デリミタ文字列内のトークンをカウントします。
```
uElements("James T. Kirk") // returns 3
```

uToken

説明 デリミタ文字列の N 番目の要素を返します。2 番目のパラメータはトークン番号を指定します。インデックスは 1 から始まります。3 番目のパラメータを省略した場合は、スペース (ASCII 32) がデリミタとして使用されます。

構文 string uToken(text, index [, delimiter])

パラメータ *string text*
調査する文字列

Integer index
返すトークンの番号

string delimiter (オプション)
使用するデリミタ。デフォルトのデリミタはスペース文字です。

例

```
uToken("James T. Kirk", 1) // returns "James"
uToken("James T. Kirk", 2) // returns "T."
```

その他

関数	説明
uCommandLine	現在のプロセスのコマンド・ライン文字列を返す。
uGetEnv	環境変数の値を返す。
uGuid	GUID (Global Unique Identifier) を返す。
uMD5	指定された文字列のチェックサムを生成する。
uScriptLoad	JavaScript をロードして評価し、結果を返す。
uSetEnv	環境変数の値を設定する。
uSetLocale	ロケールの日時設定を別の言語に変更する。
uSleep	指定されているミリ秒間、プロセスをサスペンドする。
uSystemFolder	事前に定義されているアプリケーションとシステムのパスを返す。

uCommandLine

説明　現在のプロセスのコマンド・ライン文字列を返します。

構文　string uCommandLine()

例

```
uCommandLine() // returns
"GridNode.exe --port 5124"
```

注意 uCommandLine は UNIX ではサポートされていません。

uGetEnv

説明　環境変数の値を返します。

構文　string uGetEnv(variable)

パラメータ　*string variable*

読み込む環境変数の名前

例

```
uGetEnv("LOAD_MAX_VALUE")
```

uGuid

説明 GUID (Global Unique Identifier) を返します。次のフォーマットを使用できます。

- *numeric* - 数字のみ
- *base64* - base64 のコード化
- *hex* - ハイフンなしの 16 進フォーマット

構文 string uGuid([format])

パラメータ *string format* (オプション)
返す GUID 値のフォーマット

例

```
uGuid() // returns for example A8A10D9F-963F-4914-
8D6FC8527A50EF2A
```

uMD5

説明 32 文字の固定長を持つ、指定された文字列のチェックサムを生成します。

構文 string uMD5(text)

パラメータ *string text*
チェックサムを構築するテキスト

例

```
uMD5("Austin Powers") // returns
"C679A893E3DA2CC0741AC7F527B1D4EB"
```

uScriptLoad

説明 JavaScript をロードして評価し、結果を返します。

構文 string uScriptLoad(filelocation)

パラメータ *string filelocation*
ロードする JavaScript ファイル

例 外部の JavaScript ファイルをロードします。

```
uScriptLoad("¥¥server3¥myScripts¥basicFunctions.js")
```

uSetEnv

説明　環境変数の値を設定します。

構文　string uSetEnv(variable, value)

パラメータ　*string variable*
　設定する環境変数の名前
string value
　設定する値

例

```
uSetEnv("LOAD_MAX_VALUE", IN.Date)
```

uSetLocale

説明　ロケールの日時設定を別の言語に変更します。

構文　string uSetLocale([language] [, country] [, codepage])

パラメータ　*string language* (オプション)
　使用する言語文字列 (以下のリストを参照)
string country (オプション)
　使用する国名 (以下のリストを参照)
string codepage (オプション)
　文字列としてのコード・ページ番号

例　月名を異なる言語で取得します。

```
locale:uSetLocale("English")    // switch to english
uMonthName("2005-03-22") // returns "March"
uSetLocale("German")     // switch to german
uMonthName("2005-03-22") // returns "Marz"
uSetLocale("C")          // switch back to OS default
```

使用法

言語文字列
次の言語文字列が認識されます。オペレーティング・システムがサポートしていない言語は uSetLocale によって受け入れられません。

注意 3 文字の言語文字列コードは Windows NT と Windows 95 でのみ有効です。

プライマリ言語	サブ言語	言語文字列
中国語	中国語	"chinese"
中国語	中国語 (簡体字)	"chinese-simplified" または "chs"
中国語	中国語 (繁体字)	"chinese-traditional" または "cht"
チェコ語	チェコ語	"csy" または "czech"
デンマーク語	デンマーク語	"dan" または "danish"
オランダ語	オランダ語 (ベルギー)	"belgian"、 "dutch-belgian"、 または "nlb"
オランダ語	オランダ語 (デフォルト)	"dutch" または "nld"
英語	英語 (オーストラリア)	"australian"、 "ena"、 または "english-aus"
英語	英語 (カナダ)	"canadian"、 "enc"、 または "english-can"
英語	英語 (デフォルト)	"english"
英語	英語 (ニュージーランド)	"english-nz" または "enz"
英語	英語 (英国)	"eng"、 "english-uk"、 または "uk"
英語	英語 (米国)	"english"、 "americanenglish"、 "english-american"、 "english-us"、 "english-usa"、 "enu"、 "us"、 または "usa"
フィンランド語	フィンランド語	"fin" または "finnish"
フランス語	フランス語 (ベルギー)	"frb" または "french-belgian"
フランス語	フランス語 (カナダ)	"frc" または "frenchcanadian"
フランス語	フランス語 (デフォルト)	"fra" または "french"
フランス語	フランス語 (スイス)	"french-swiss" または "frs"
ドイツ語	ドイツ語 (オーストリア)	"dea" または "germanaustrian"
ドイツ語	ドイツ語 (デフォルト)	"deu" または "german"
ドイツ語	ドイツ語 (スイス)	"des"、 "german-swiss"、 または "swiss"
ギリシャ語	ギリシャ語	"ell" または "greek"
ハンガリー語	ハンガリー語	"hun" または "hungarian"
アイスランド語	アイスランド語	"icelandic" または "isl"
イタリア語	イタリア語 (デフォルト)	"ita" または "italian"
イタリア語	イタリア語 (スイス)	"italian-swiss" または "its"

プライマリ言語	サブ言語	言語文字列
日本語	日本語	"japanese" または "jpn"
韓国語	韓国語	"kor" または "korean"
ノルウェー語	ノルウェー語 (ブークモール)	"nor" または "norwegianbokmal"
ノルウェー語	ノルウェー語 (デフォルト)	"norwegian"
ノルウェー語	ノルウェー語 (ニーノシュク)	"non" または "norwegiannynorsk"
ポーランド語	ポーランド語	"plk" または "polish"
ポルトガル語	ポルトガル語 (ブラジル)	"portuguese-brazilian" または "ptb"
ポルトガル語	ポルトガル語 (デフォルト)	"portuguese" または "ptg"
ロシア語	ロシア語 (デフォルト)	"rus" または "russian"
スロバキア語	スロバキア語	"sky" または "slovak"
スペイン語	スペイン語 (デフォルト)	"esp" または "spanish"
スペイン語	スペイン語 (メキシコ)	"esm" または "spanish-mexican"
スペイン語	スペイン語 (現代)	"esn" または "spanish-modern"
スウェーデン語	スウェーデン語	"sve" または "swedish"
トルコ語	トルコ語	"trk" または "turkish"

国／地域設定

次のリストは、uSetLocale によって認識される国／地域文字列を示します。オペレーティング・システムがサポートしていない国／地域の文字列は uSetLocale によって受け入れられません。3 文字の国／地域名コードは ISO/IEC (International Organization for Standardization, International Electrotechnical Commission) の仕様 3166 に基づいています。

国／地域	国／地域文字列
オーストラリア	"aus" または "australia"
オーストリア	"austria" または "aut"
ベルギー	"bel" または "belgium"
ブラジル	"bra" または "brazil"
カナダ	"can" または "canada"
チェコ	"cze" または "czech"
デンマーク	"denmark" または "dnk"
フィンランド	"fin" または "finland"
フランス	"fra" または "france"

国／地域	国／地域文字列
ドイツ	"deu" または "germany"
ギリシャ	"grc" または "greece"
香港特別行政区	"hkg"、 "hong kong"、 または "hong-kong"
ハンガリー	"hun" または "hungary"
アイスランド	"iceland" または "isl"
アイルランド	"ireland" または "irl"
イタリア	"ita" または "italy"
日本	"japan" または "jpn"
韓国	"kor" または "korea"
メキシコ	"mex" または "mexico"
オランダ	"nld"、 "holland"、 または "netherlands"
ニュージーランド	"new zealand"、 "new-zealand"、 "nz"、 または "nzl"
ノルウェー	"nor" または "norway"
中華人民共和国	"china"、 "chn"、 "pr china"、 または "pr-china"
ポーランド	"pol" または "poland"
ポルトガル	"prt" または "portugal"
ロシア	"rus" または "russia"
シンガポール	"sgp" または "singapore"
スロバキア	"svk" または "slovak"
スペイン	"esp" または "spain"
スウェーデン	"swe" または "sweden"
スイス	"che" または "switzerland"
台湾	"taiwan" または "twn"
トルコ	"tur" または "turkey"
英国	"britain"、 "england"、 "gbr"、 "great britain"、 "uk"、 "united kingdom"、 または "united-kingdom"
アメリカ合衆国	"america"、 "united states"、 "unitedstates"、 "us"、 または "usa"

uSleep

説明　指定されているミリ秒間、プロセスをサスペンドします。

構文　string uSleep(msecs)

パラメータ　*integer msecs*
サスペンドするミリ秒数

例
```
uSleep(1000) // suspends the process for one second
```

uSystemFolder

説明　事前に定義されているアプリケーションとシステムのパスを返します。

構文　string uSystemFolder([foldertype])

例
```
uSystemFolder("APP_LOG") // returns the path to the log
directory
```

使用法　uSystemFolder 関数は、ファイルシステム上の特殊なディレクトリにアクセスするために使用できます。次のフォルダを指定できます。

グループ	名前	説明
アプリケーション	APP_MAIN	基本アプリケーション・パス。一般的なパスは *C:¥Program Files¥ETL*。
アプリケーション	APP_LIB	共有ライブラリ・ディレクトリ。通常はアプリケーション・ディレクトリの lib フォルダにある。
アプリケーション	APP_LOG	共有ライブラリ・ディレクトリ。通常はアプリケーション・ディレクトリの lib フォルダにある。
アプリケーション	APP_CONFIG	設定ファイル・ディレクトリ。通常はアプリケーション・ディレクトリの etc フォルダにある。
アプリケーション	APP_LICENSE	ライセンス・ファイル・ディレクトリ。通常はアプリケーション・ディレクトリの license フォルダにある。
アプリケーション	APP_SCRIPT	スクリプト・ディレクトリ。通常はアプリケーション・ディレクトリの scripts フォルダにある。
アプリケーション	APP_GRAMMAR	文法ディレクトリ。通常はアプリケーション・ディレクトリの grammar フォルダにある。
アプリケーション	APP_LANGUAGE	言語ファイル・ディレクトリ。通常はアプリケーション・ディレクトリの language フォルダにある。
アプリケーション	APP_DATABASE	データベース・ディレクトリ。通常はアプリケーション・ディレクトリの database フォルダにある。
アプリケーション	APP_TEMP	テンポラリ・ディレクトリ。通常はアプリケーション・ディレクトリの temp フォルダにある。

グループ	名前	説明
アプリケーション	APP_DEMODATA	デモデータ・ディレクトリ。通常はアプリケーション・ディレクトリの demodata フォルダにある。
アプリケーション	APP_USERDATA	ユーザ固有のファイルが格納されるディレクトリ。一般的なパスは *C:¥Documents and Settings¥username¥Application Data¥ETL*。
Windows	ALTSTARTUP	ユーザのローカライズされていない [スタートアップ] プログラム・グループに対応するファイル・システム・ディレクトリ。
Windows	APPDATA	アプリケーション固有データの共通リポジトリとして機能するファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥username¥Application Data*。
Windows	CDBURN_AREA	CD への書き込みを待機しているファイルの作業領域として機能するファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥username¥Local Settings¥Application Data¥Microsoft¥CD Burning*。
Windows	COMMON_ADMINTOOLS	コンピュータの全ユーザ用の管理ツールを含むファイル・システム・ディレクトリ。
Windows	COMMON_APPDATA	全ユーザ用のアプリケーション・データを含むファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥All Users¥Application Data*。
Windows	COMMON_DESKTOPDIRECTORY	全ユーザ用にデスクトップに表示されるファイルとフォルダを含むファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥All Users¥Desktop*。Windows NT システムでのみ有効。
Windows	COMMON_DOCUMENTS	全ユーザに共通のドキュメントを含むファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥All Users¥Documents*。
Windows	COMMON_FAVORITES	全ユーザに共通のお気に入り項目用共通リポジトリとして機能するファイル・システム・ディレクトリ。Windows NT システムでのみ有効。
Windows	COMMON_MUSIC	全ユーザに共通の音楽ファイル用リポジトリとして機能するファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥All Users¥Documents¥My Music*。
Windows	COMMON_PICTURES	全ユーザに共通のイメージ・ファイル用リポジトリとして機能するファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥All Users¥Documents¥My Pictures*。

グループ	名前	説明
Windows	COMMON_PROGRAMS	全ユーザの [スタート] メニューに表示される共通プログラム・グループ用ディレクトリを含むファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥All Users¥Start Menu¥Programs*。Windows NT システムでのみ有効。
Windows	COMMON_STARTMENU	全ユーザに対して [スタート] メニューに表示されるプログラムとフォルダを含むファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥All Users¥Start Menu*。Windows NT システムでのみ有効。
Windows	COMMON_STARTUP	全ユーザに対して Startup フォルダに表示されるプログラムを含むファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥All Users¥Start Menu¥Programs¥Startup*。Windows NT システムでのみ有効。
Windows	COMMON_TEMPLATES	全ユーザが使用できるテンプレートを含むファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥All Users¥Templates*。Windows NT システムでのみ有効。
Windows	COMMON_VIDEO	全ユーザに共通のビデオ・ファイル用リポジトリとして機能するファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥All Users¥Documents¥My Videos*。
Windows	COOKIES	インターネットのクッキー用共通リポジトリとして機能するファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥username¥Cookies*。
Windows	DESKTOP	ネームスペースのルートである Windows デスクトップを表す仮想フォルダ。
Windows	DESKTOPDIRECTORY	デスクトップ上のファイル・オブジェクトを物理的に格納するためのファイル・システム・ディレクトリ (デスクトップ・フォルダ自体と混合しないでください)。一般的なパスは *C:¥Documents and Settings¥username¥Desktop*。
Windows	FAVORITES	ユーザのお気に入り項目用共通リポジトリとして機能するファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥username¥Favorites*。
Windows	FONTS	フォントを含む仮想フォルダ。一般的なパスは *C:¥Windows¥Fonts*。
Windows	HISTORY	インターネットの履歴項目用共通リポジトリとして機能するファイル・システム・ディレクトリ。

グループ	名前	説明
Windows	INTERNET_CACHE	テンポラリ・インターネット・ファイル用共通リポジトリとして機能するファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥username¥Local Settings¥Temporary Internet Files*。
Windows	MYDOCUMENTS	マイ・ドキュメントのデスクトップ項目を表す仮想フォルダ。
Windows	MYMUSIC	音楽ファイル用の共通リポジトリとして機能するファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥User¥My Documents¥My Music*。
Windows	MYPICTURES	イメージ・ファイル用の共通リポジトリとして機能するファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥username¥My Documents¥My Pictures*。
Windows	MYVIDEO	ビデオ・ファイル用の共通リポジトリとして機能するファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥username¥My Documents¥My Videos*。
Windows	NETHOOD	[マイ ネットワーク] 仮想フォルダに存在できるリンク・オブジェクトを含むファイル・システム・ディレクトリ。これは、ネットワーク・ネームスペース・ルートを表す *CSIDL_NETWORK* とは異なる。一般的なパスは *C:¥Documents and Settings¥username¥NetHood*。
Windows	PERSONAL	マイ・ドキュメントのデスクトップ項目を表す仮想フォルダ。これは MYDOCUMENTS と同等。
Windows	PRINTHOOD	[プリンタ] 仮想フォルダに存在できるリンク・オブジェクトを含むファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥username¥PrintHood*。
Windows	PROFILE	ユーザのプロファイル・フォルダ。一般的なパスは *C:¥Documents and Settings¥username*。アプリケーションではこのレベルでファイルやフォルダを作成しないでください。ファイルやフォルダのデータは *APPDATA* または *LOCAL_APPDATA* によって参照されているロケーションの下に配置してください。
Windows	PROGRAM_FILES	プログラム・ファイル・フォルダ。一般的なパスは *C:¥Program Files*。

グループ	名前	説明
Windows	PROGRAM_FILES_COMMON	アプリケーション間で共有されるコンポーネント用フォルダ。一般的なパスは *C:¥Program Files¥Common*。Windows NT、Windows 2000、Windows XP システムでのみ有効。
Windows	PROGRAMS	ユーザのプログラム・グループ (これら自体はファイル・システム・ディレクトリ) を含むファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥username¥Start Menu¥Programs*。
Windows	RECENT	ユーザが最も最近使用したドキュメントへのショートカットを含むファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥username¥My Recent Documents*。このフォルダにショートカットを作成するには、*SHAddToRecentDocs* を使用してください。ショートカットの作成に加え、この関数は、最近使ったドキュメントのシェルのリストを更新し、ショートカットを [スタート] メニューの [最近使ったファイル] サブメニューに追加します。
Windows	SENDTO	[送信] メニュー項目を含むファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥username¥SendTo*。
Windows	STARTMENU	[スタート] メニュー項目を含むファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥username¥Start Menu*。
Windows	STARTUP	ユーザの [スタートアップ] プログラム・グループに対応するファイル・システム・ディレクトリ。システムは、ユーザが Windows NT にログオンするか、Windows 95 を起動したときにこれらのプログラムを起動する。一般的なパスは *C:¥Documents and Settings¥username¥Start Menu¥Programs¥Startup*。
Windows	SYSTEM	Windows システム・フォルダ。一般的なパスは *C:¥Windows¥System32*。
Windows	TEMPLATES	ドキュメント・テンプレート用共通リポジトリとして機能するファイル・システム・ディレクトリ。一般的なパスは *C:¥Documents and Settings¥username¥Templates*。
Windows	WINDOWS	Windows ディレクトリまたは SYSROOT。これは *%windir%* または *%SYSTEMROOT%* 環境変数に対応している。一般的なパスは *C:¥Windows*。

ネットワーク

関数	説明
uHostname	ローカル・ネットワーク名を返す。
uSMTP	メールを SMTP サーバに送信する。

uHostname

説明 ローカル・ネットワーク名を返します。

構文 string uHostname()

例

```
uHostname()  // returns something like "pollux" or
"castor"
```

uSMTP

説明 メールを SMTP サーバに送信します。

構文

bool uSMTP(serverURL, sender, recipients, subject, body)
bool uSMTP(sender, recipients, subject, body)
bool uSMTP(recipients, subject, body)bool uSMTP(subject, body)
bool uSMTP(body)

パラメータ

string serverURL
使用する SMTP サーバ、ポート、ユーザ名、パスワードを指定した URL

string sender
送信側の電子メール・アドレス

string recipients
カンマで区切られた受信者のリスト

string subject
メール・メッセージの件名

string body
電子メール・メッセージの内容

例

```
uSMTP("Just a mail")

uSMTP("Testmail!", "Just a mail")
```

使用法

uSMTP 関数は、SMTP サーバを使用して複数の受信者に電子メールを送信できるようにします。

SMTP サーバ、ポート、ユーザ、パスワードの指定

SMTP サーバを指定するサーバ URL の構文は次のとおりです。

```
protocol://user:password@server:port
```

プロトコルには次のいずれかを使用できます。
< 空 > ― 該当する場合は SSL 暗号化を使用した SMTP

SMTP ― SSL 暗号化を使用しない SMTP

SMTPS ― SSL 暗号化を使用した SMTP

ユーザとパスワード ― ユーザ名とパスワードはクライアントを認証するために使用されます。これらを指定しない場合、認証は行われません。

注意 ユーザ名に @ 記号が含まれている場合は、曖昧になるのを避けるために # に置き換えてください。

ポート ― 使用する TCP ポート。デフォルトは 25 です。

URL の例

```
myServer
myServer:123
SMTPS://myServer:123
Me:secret@myServer
```

受信者の指定 ― カンマで区切ったアドレスのリストを追加できます。デフォルトでは、すべての受信者が直接指定されます。CC または BC を指定するには、メール・アドレスにプレフィクスを追加します。

```
user@host.domain
My Name <user@host.domain>
To: My Name <user@host.domain>
To: user@host.domain
Cc: My Name <user@host.domain>
To: user@host.domain, Bcc: Test User
<test@myserver.com>
```

セキュリティ ― SMTP サーバが暗号化された接続を使用した通信を許可している場合、これは自動的に行われます。ユーザ名とパスワードが指定されている場合、次の順序で認証方法が試されます：PLAIN、LOGIN。

カスタム・デフォルト － 個人用のデフォルトを INI ファイルに指定できます。

```
[SMTP]
ServerURL=<your default server URL>
Sender=<your default sender>
Recipients=<your default recipients>
Subject=<your default subject>
```

INI ファイルの例：

```
[SMTP]
ServerURL=maxm:secret@mail.gmail.com
Sender= Maxi <Max.Mustermann@ gmail.com>
Recipients=ETLAdmin@MyCompany.com, Cc:  QA
qa@MyCompany.com
Subject=ETL Message
```

数値

関数	説明
uAbs	正または負の記号を無視して実数の大きさを返す。
uCeil	引数以上の値を持つ最小整数を返す。
uDiv	整数除算結果を返す。
uExp	指数関数の底 e を返す。
uFloor	引数以下の値を持つ最大整数を返す。
uLn	数値の自然対数 (底 e) を返す。
uLog	数値の対数を返す。
uMod	除算のモジュロを返す。
uPow、uPower	指定された累乗になる基数式の値を返す。
uRandom	乱数を返す。
uRound	最も近い整数に丸められた引数を返す。
uSgn	指定された値の符号を返す。
uSqrt	指定された値の平方根を返す。

uAbs

説明　正または負の記号を無視して実数の大きさを返します。

構文　number uAbs(value)

パラメータ　*number value*
計算する数

例
```
uAbs(1522)       // returns 1522
uAbs('-123.45') // returns 123.45
uAbs('123ABC')   // returns 0
```

uCeil

説明　引数以上の値を持つ最小整数を返します。

構文　number uCeil(value)

パラメータ　*number value*
計算する数

例　数を丸めます。
```
uCeil(1523.1) // returns 1524
uCeil(1523.9) // returns 1524
```

uDiv

説明　整数除算結果を返します。

構文　number uDiv(value, divisor)

パラメータ　*number value*
計算する数
number divisor
除算する数

例
```
uDiv(10, 3) // returns 3
```

uExp

説明 指数関数の底 e を返します。

構文 number uExp(value)

パラメータ *number value*
計算する数

例
```
uExp(1) // returns "2.718281828459045"
```

uFloor

説明 引数以下の値を持つ最大整数を返します。

構文 number uFloor(value)

パラメータ *number value*
計算する数

例
```
uFloor(1523.1) // returns 1523
uFloor(1523.9) // returns 1523
```

uLn

説明 数値の自然対数 (底 e) を返します。

構文 number uLn(value)

パラメータ *number value*
計算する数

例
```
uLn(2.718281828) // returns 0.999999
```

uLog

説明　数値の対数を返します。

構文　number uLog(value [, base])

パラメータ　*number value*
　計算する数
number base (オプション)
　対数の底。省略した場合は、10 の底が使用されます。

例
```
uLog(100) // returns 2
uLog(16, 2) // returns 4
```

uMod

説明　除算のモジュロを返します。

構文　number uMod(value, divisor)

パラメータ　*number value*
　計算する数
number divisor
　除算する数

例
```
uMod(10, 3) // returns 1
```

uPow、uPower

説明　指定された累乗になる基数式の値を返します。

構文　number uPow(value, exponent)

パラメータ　*number value*
　計算する数
number exponent
　指数として使用される数

例
```
uPow(10, 3) // returns 1000
```

uRandom

説明 乱数を返します。

構文 number uRandom()

例 乱数

```
uRandom() // returns a value like "0.696654639123727"
```

uRound

説明 最も近い整数に丸められた引数を返します。

構文 number uRound(value [, scale])

パラメータ *number value*
計算する数
number scale (オプション)
桁数

例

```
uRound(10.1) // returns "10"
uRound(10.49) // returns "10"
uRound(10.5) // returns "11"
uRound(10.9) // returns "11"
uRound(1.235, 2) // returns "1.24"
```

uSgn

説明 指定された値の符号を返します。

構文 number uSgn(value)

パラメータ *number value*
計算する数

例

```
uSgn(-10.4) // returns -1
uSgn(0)     // returns 0
uSgn(10.4)  // returns 1
uSgn(null)  // returns null
```

uSqrt

説明　指定された値の平方根を返します。

構文　number uSqrt(value)

パラメータ　*number value*
　計算する数

例

```
uSqrt(25)   // returns 5
uSqrt(0)    // returns 0
uSqrt(null) // returns null
```

スクリプト

関数	説明
uEvaluate	関数または JavaScript 式を評価し、結果を返す。

uEvaluate

説明　関数または JavaScript 式を評価し、結果を返します。

構文　string uEvaluate(expression)

パラメータ　*Number expression*
　評価する JavaScript コード

例

評価関数式：

```
uEvaluate("3 + 5")
uEvaluate("parseFloat(IN.Salary) + 1500")
```

カスタム関数の定義：

```
uEvaluate("function timesTwo(a){ return a*2; }")
```

カスタム関数の使用：

```
uEvaluate("timesTwo(4)")
uEvaluate("timesTwo(IN.Salary)")
```

スクリプトの評価：

```
uEvaluate("if ("parseFloat(IN.Salary) > 2000") {2000;}
else {("parseFloat(IN.Salary) + 500");}")
```

文字列

関数	説明
uAsc、uUnicode	指定された文字の Unicode 文字値を返す。
uChr、uUniChr	指定された数に対応する Unicode 文字列を返すか、文字列をフォーマットする。
uCap	頭文字を大文字に変換した文字列を返す。
uCon、uConcat	指定されたパラメータすべてを単一の文字列に連結する。
uJoin	特殊な null および空の値処理を使用して、デリミタ文字列を連結する。
uLeft	文字列の左端の N 文字を返す。
uLength、uLen	文字列の長さを返す。
uSubstr、uMid	文字列の一部を返す。
uLPos	文字列内の部分文字列の最初の位置を見つける。
uLower、uLow	小文字に変換した入力文字列を返す。
uLStuff	文字列の左端を指定された長さまで補充する。
uLTrim	文字列の左端から文字を削除する。
uRepeat	指定された文字列を N 回繰り返したものを返す。
uReplace	文字列の一部を置換する。
uReverse	文字列を逆にする。
uRight	文字列の右端の N 文字を返す。
uRPos	文字列内の部分文字列の最後の位置を見つける。
uRStuff	文字列の右端を指定された長さまで補充する。
uRTrim	文字列の右端から文字を削除する。
uTrim	文字列の両側から文字を削除する。
uUpper、uUpp	大文字に変換した入力文字列を返す。

uAsc、uUnicode

説明　指定された文字の Unicode 文字値を返します。

構文　number uAsc(value [, index])

パラメータ　*string value*
入力文字列
number index (オプション)
ASCII 値を読み込む文字位置

例
```
uAsc("Big Ben")    // returns 66
uAsc("Big Ben", 2) // returns 105
```

uChr、uUniChr

説明　指定された数に対応する Unicode 文字列を返すか、文字列をフォーマットします。

構文　string uChr(params, ...)

パラメータ　*params*
式または値のリスト

例
```
uChr(64) // returns "@"
uChr("¥u0064¥u006f¥u0067") // returns "dog"
uChr(65,"pple") // returns "apple"
```

uCap

説明　頭文字を大文字に変換した文字列を返します。つまり、文字列内の各単語の最初の文字が大文字になります。

構文　string uCap(text)

パラメータ　*Input text*
先頭を大文字にする文字列

例
```
uCap('fArmeR, ASTROnaut') // returns 'Farmer,
Astronaut'
uCap('the first weekend') // returns 'The First Weekend'
```

uCon、uConcat

説明 指定されたパラメータすべてを単一の文字列に連結します。

構文 string uConcat(params)

パラメータ *params*
任意のデータ型の式または値のリスト

例
```
uConcat("For ", 3, " years.") returns "For 3 years."
```

uJoin

説明 特殊な null および空の値処理を使用して、デリミタ文字列を連結します。

構文 string uJoin(delimiter, allowEmpty, params, ...)

パラメータ *string delimiter*
他のすべての文字列部分間に使用するデリミタ

number allowEmpty
空のフィールドを許可するかどうかを示すフラグ (0 ／ 1)

string params
連結する文字列のリスト

例
```
uJoin("-", 1, "James", "", "Tiberius", "Kirk") //
returns "James--Tiberius-Kirk"

uJoin("-", 0, "James", "", "Tiberius", "Kirk") //
returns "James-Tiberius-Kirk"
```

uLeft

説明 文字列の左端の N 文字を返します。

構文 string uLeft(input, chars)

パラメータ *string input*
入力文字列

number chars
取得する文字列の数

例
```
uLeft("James T. Kirk", 5) // returns "James"

uLeft(null, 5)            // returns null
```

uLength、uLen

説明　文字列の長さを返します。

構文　number uLength(input)

パラメータ　*string input*
入力文字列

例
```
uLength("James T. Kirk") // returns 13
```

uSubstr、uMid

説明　文字列の一部を返します。

構文　string uSubstr(input, position, length)

パラメータ　*string input*
入力文字列
number position
読み込みの開始位置
number length
読み込む文字数

例
```
uSubstr("James T. Kirk", 7, 2) // returns "T."
```

uLPos

説明　文字列内の部分文字列の最初の位置を見つけます。0 の結果は、部分文字列が見つからなかったことを示します。

構文　string uLPos(input, substring)

パラメータ　*string input*
入力文字列
string substring
検索する部分文字列

例
```
uLPos("James T. Kirk", "T") //returns 7
```

uLower、uLow

説明 小文字に変換した入力文字列を返します。

構文 string uLower(input)

パラメータ *string input*
変換する文字列

例
```
uLower("James T. Kirk") // returns "james t. kirk"
```

uLStuff

説明 文字列の左端を指定された長さまで補充します。

構文 string uLStuff(input, length [, stuff])

パラメータ *string input*
補充する文字列
number length
文字列の新しい長さ
string stuff (オプション)
追加する文字列。デフォルトは空のスペース (ASCII 32) です。

例
```
uLStuff("3.5", 5)      // returns "  3.5"
uLStuff("3.5", 5, "0") // returns "003.5"
```

uLTrim

説明 文字列の左端から文字を削除します。2 番目のパラメータが省略された場合は、デフォルトでスペース (ASCII 32) が想定されます。

構文 string uLTrim(input, trimstring)

パラメータ *string input*
トリミングされる文字列
string trimstring
トリミングする文字列

例
```
uLTrim(" 3.5")       // returns "3.5"
uLTrim("003.5", "0") // returns "3.5"
```

uRepeat

説明　指定された文字列を N 回繰り返したものを返します。

構文　string uRepeat(input, repeats)

パラメータ

string input
　繰り返す文字列
number repeats
　入力文字列を繰り返す回数

例

```
uRepeat("Hello ", 4) // returns "Hello Hello Hello Hello "
```

uReplace

説明　文字列の一部を置換します。

構文　string uReplace(input, search, replace)

パラメータ

string input
　操作する文字列
string search
　検索パターン
string replace
　一致したものを置換する文字列

例

```
uReplace("At four o' clock he became four", "four", "4")
// returns "At 4 o' clock he became 4"
```

uReverse

説明　文字列を逆にします。

構文　string uReverse(input)

パラメータ

string input
　逆にする文字列

例

```
uReverse("Smith")  // returns "htimS"
```

uRight

説明 文字列の右端の N 文字を返します。

構文 string uRight(input, chars)

パラメータ

string input
入力文字列

number chars
読み込む文字数

例

```
uRight("James T. Kirk", 4) // returns "Kirk"
uRight(null, 5) / / returns null
```

uRPos

説明 文字列内の部分文字列の最後の位置を見つけます。

構文 string uRPos(input, substring)

パラメータ

string input
入力文字列

string substring
検索する部分文字列

例 部分文字列の最後のオカレンスを検索します。

```
uRPos("James T. Kirk", "T") //returns 7
```

uRStuff

説明 文字列の右端を指定された長さまで補充します。

構文 string uRStuff(input, length [, stuffstring])

パラメータ

string input
入力文字列

number length
結果文字列の新しい長さ

string stuffstring (オプション)
追加する文字列

例

```
uRStuff("3.5", 5)      // returns "3.5  "
uRStuff("3.5", 5, "0") // returns "3.500"
```

uRTrim

説明　文字列の右端から文字を削除します。

構文　string uRTrim(input [, trimstring])

パラメータ　*string input*
　入力文字列
string trimstring (オプション)
　トリミングする文字列

例
```
uRTrim("3.5 ")        // returns "3.5"
uRTrim("3.500", "0") // returns "3.5"
```

uTrim

説明　文字列の両側から文字を削除します。

構文　string uTrim(input [, trimstring])

パラメータ　*string input*
　入力文字列
string trimstring (オプション)
　トリミングする文字列

例
```
uTrim(" 3.5 ")        // returns "3.5"
uTrim("003.500", "0") // returns "3.5"
```

uUpper、uUpp

説明　大文字に変換した入力文字列を返します。

構文　string uUpper(input)

パラメータ　*string input*
　入力文字列

例
```
uUpper("James T. Kirk") // returns "JAMES T. KIRK"
```

演算子

関数	説明
uEQ	両方のパラメータが等しく、NULL のパラメータがない場合に 1 を返す。
uNE	両方のパラメータが異なり、NULL のパラメータがない場合に 1 を返す。
uGT	最初のパラメータが 2 番目のパラメータより大きく、NULL のパラメータがない場合に 1 を返す。
uGE	最初のパラメータが 2 番目のパラメータより大きいか等しく、NULL のパラメータがない場合に 1 を返す。
uLT	最初のパラメータが 2 番目のパラメータより小さく、NULL のパラメータがない場合に 1 を返す。
uLE	最初のパラメータが 2 番目のパラメータより小さいか等しく、NULL のパラメータがない場合に 1 を返す。

注意 互換性を保つために、演算子は関数としても提供されます。

uEQ

説明 両方のパラメータが等しく、NULL のパラメータがない場合に 1 を返します。

構文 number uEQ(value1, value2)

パラメータ *value1, value2*

比較する数値または文字列値

例

```
uEQ(1,2)     // returns 0
uEQ(1,1)     // returns 1
uEQ(null,1) // returns 0
```

uNE

説明 両方のパラメータが異なり、NULL のパラメータがない場合に 1 を返します。

構文 number uNE(value1, value2)

パラメータ　*value1, value2*
比較する数値または文字列値

例
```
uNE(1,2)    // returns 1
uNE(1,1)    // returns 0
uNE(null,1) // returns 0
```

uGT

説明　最初のパラメータが 2 番目のパラメータより大きく、NULL のパラメータがない場合に 1 を返します。

構文　number uGT(value1, value2)

パラメータ　*value1, value2*
比較する数値または文字列値

例
```
uGT(2,1)    // returns 1
uGT(1,2)    // returns 0
uGT(1,1)    // returns 0
uGT(null,1) // returns 0
```

uGE

説明　最初のパラメータが 2 番目のパラメータより大きいか等しく、NULL のパラメータがない場合に 1 を返します。

構文　number uGE(value1, value2)

パラメータ　*value1, value2*
比較する数値または文字列値

例
```
uGE(2,1)    // returns 1
uGE(1,2)    // returns 0
uGE(1,1)    // returns 1
uGE(null,1) // returns 0
```

uLT

説明 最初のパラメータが 2 番目のパラメータより小さく、NULL のパラメータがない場合に 1 を返します。

構文 number uLT(value1, value2)

パラメータ *value1, value2*

比較する数値または文字列値

例

```
uLT(2,1)     // returns 0
uLT(1,2)     // returns 1
uLT(1,1)     // returns 0
uLT(null,1) // returns 0
```

uLE

説明 最初のパラメータが 2 番目のパラメータより小さいか等しく、NULL のパラメータがない場合に 1 を返します。

構文 number uLE(value1, value2)

パラメータ *value1, value2*

比較する数値または文字列値

例

```
uLE(2,1)     // returns 0
uLE(1,2)     // returns 1
uLE(1,1)     // returns 1
uLE(null,1) // returns 0
```

三角法

関数	説明
uAcos	数のアーク・コサインをラジアンで返す。
uAsin	数のアーク・サインをラジアンで返す。
uAtan	数のアーク・タンジェントをラジアンで返す。
uCos	数のコサインをラジアンで返す。
uSin	数のサインをラジアンで返す。
uTan	数のタンジェントをラジアンで返す。

uAcos

説明　数のアーク・コサインをラジアンで返します。
構文　number uAcos(value)
パラメータ　*number value*
　入力値

uAsin

説明　数のアーク・サインをラジアンで返します。
構文　number uAsin(value)
パラメータ　*number value*
　入力値

uAtan

説明　数のアーク・タンジェントをラジアンで返します。
構文　number uAtan(value)
パラメータ　*number value*
　入力値

uCos

説明　数のコサインをラジアンで返します。
構文　number uCos(value)
パラメータ　*number value*
　入力値

uSin

説明 数のサインをラジアンで返します。

構文 number uSin(value)

パラメータ *number value*
入力値

uTan

説明 数のタンジェントをラジアンで返します。

構文 number uTan(value)

パラメータ *number value*
入力値

付 録 B

接続パラメータ

この付録では、データベース設定オプションについて説明します。また、一部のサポートされているインタフェースに関する追加情報も提供します。

インタフェース固有のデータベース・オプション

DB オプション	インタフェースとデフォルト値						説明
	Oracle	ODBC/DB2	Sybase 15	Sybase	OLEDB	SQLite Persistent	
Allow fallback	1 (使用しない)	1 (使用しない)	1	1	1 (使用しない)	1 (使用しない)	1 に設定すると、バルク操作を使用できない場合、標準のロード操作が使用されます。
Always use logon credentials	使用不可	0	使用不可	使用不可	使用不可	使用不可	ODBC 接続文字列を作成する場合は、必ずクレデンシャルを接続文字列に追加します。
API trace	使用不可	使用不可	False	False	使用不可	使用不可	CTLIB トレース機能を有効にします。
API version	使用不可	使用不可	150	125	使用不可	使用不可	CTLIB API バージョンの互換性。

DB オプション	インタフェースとデフォルト値						説明
	Oracle	ODBC/DB2	Sybase 15	Sybase	OLEDB	SQLite Persistent	
Auto commit	使用不可	1 データベースが IQ であるか、または ODBC ドライバが ASA ODBC 9 である場合、ODBC はユーザ入力ではなく 0 を使用します。	0	0	使用不可	使用不可	接続をオート・コミット・モードにします。
Auto vacuum	使用不可	使用不可	使用不可	使用不可	使用不可	0	オブジェクトがデータベースから削除されたときに、領域を再利用します。
Block inserts	使用不可	auto ODBC ドライバが ASA9 であり、値が auto であり、LOB 型がなく、“Write rejected records to file” オプションが null である場合にのみ、バルク操作が使用されます。	使用不可	使用不可	使用不可	使用不可	SQL 文の実行準備を行う場合、ブロックごとのバインドを使用します。
BLOB chunk size	1024 LOB データをファイルからフェッチしている間、ETL は毎回 1024 バイトをファイルからフェッチし、データベースに書き込みます。	使用不可	使用不可	使用不可	使用不可	使用不可	LOB をトランケートするサイズを決定します。
BLOB fetch mode	LOB_INLINE	INLINE	使用不可	使用不可	使用不可	使用不可	BLOB データは、セカンダリ・ファイルに書き込まれるか、メモリ内に保持されます。INLINE に設定すると、データはメモリ内に保持されます。FILE に設定すると、データは一時的にディスクに書き込まれます。

DB オプション	インタフェースとデフォルト値						説明
	Oracle	ODBC/DB2	Sybase 15	Sybase	OLEDB	SQLite Persistent	
Busy timeout	使用不可	使用不可	使用不可	使用不可	使用不可	10	ロックされたデータベース・テーブルを検出した場合に指定の秒数だけ待機するハンドラを作成します。
Cache size	使用不可	使用不可	使用不可	使用不可	使用不可	3000	キャッシュで使用するページ数。
CLIENT_ CHARSET	使用不可	使用不可	-	-	使用不可	使用不可	CTLIB で使用する文字セット (ユーザ定義)。
Client Conversion	使用不可	使用不可	0 (ASE) 1 (ASA/IQ)	0 (ASE) 1 (ASA/IQ)	使用不可	使用不可	クライアント・ライブラリでデータを適切な形式に変換するかどうかを制御します。
Connect timeout	0 (使用しない)	0 (使用しない)	0	0	10	0	接続タイムアウトの秒数が経過すると、接続の試行を中止します。0 に設定すると、接続はタイムアウトしません。
CONVERTER_ CHARSET	使用不可	使用不可	-	-	使用不可	使用不可	CLIENT_ CONVERSION が 1 である場合に使用される文字セット。
Database name	使用不可	使用不可	-	-	使用不可	使用不可	データベースの名前。
DBMS_VER	使用不可	使用不可	-	-	使用不可	使用不可	データベースのバージョン。
Default cache size	使用不可	使用不可	使用不可	使用不可	使用不可	3000	キャッシュで使用するデフォルトのページ数。
Disconnect timeout	使用不可	10 Windows 32 ビットの場合、ETL では常にデフォルト値を使用します。 その他のプラットフォームの場合、このオプションは使用されません。	使用不可	使用不可	使用不可	使用不可	切断を試行してから *n* 秒間データベースから応答がない場合は、データベースから切断します。

DB オプション	インタフェースとデフォルト値						説明
	Oracle	ODBC/DB2	Sybase 15	Sybase	OLEDB	SQLite Persistent	
Enable bulk load	1 (使用しない)	1 ／使用しない	1	1	1 (使用しない)	1	1 に設定すると、適用可能な場合は、バルク操作が使用されます。
Enable SQL Server fast load	使用不可	使用不可	使用不可	使用不可	1	使用不可	1 に設定すると、MS SQL Server の高速ロード機能が有効になります。0 に設定すると、この機能は無効になります。
Execution timeout	0 (使用しない)	0 ／使用しない	-1	-1	使用不可	0	指定の時間 (単位は秒) が経過すると、コンポーネントの実行が停止されます (0 以下はタイムアウトがないことを意味します)。
Extended connect options	使用不可	-	使用不可	使用不可	-	使用不可	追加のドライバ固有パラメータを ODBC 接続文字列に追加できるようにします。
Full column names	使用不可	使用不可	使用不可	使用不可	使用不可	1	1 に設定すると、<table-name/alias> <column-name> のパターンに従って、カラム名が完全に修飾されます。
Internal database	使用不可	使用不可	使用不可	使用不可	使用不可	-	データベースの参照。
Isolation level	DEFAULT (使用しない)	DEFAULT	DEFAULT (使用しない)	DEFAULT (使用しない)	使用不可	DEFAULT (使用しない)	あるトランザクションが他のトランザクションによって行われたリソースまたはデータの変更から独立している必要がある度合いを定義します。
Lock resultset data	0 (使用しない)	0	0 (使用しない)	0 (使用しない)	使用不可	0	クエリ・テーブルがロックされます。これを使用して、選択したレコード・セットでプロセスが動作している間、そのセットにデータが書き込まれないようにします。選択したレコード・セットは、そのセットの最後のレコードがフェッチされると解放されます。

DB オプション	インタフェースとデフォルト値						説明
	Oracle	ODBC/DB2	Sybase 15	Sybase	OLEDB	SQLite Persistent	
Log SQL Statements to a file	0	0	0	0	0	0	1 に設定すると、すべての SQL 文はログ・ファイルまたは *SQL.log* ファイルに記録されます。
Numeric support	使用不可	0 DBMS が IQ であるか、または ODBC ドライバが ASA9 である場合、この値はユーザ入力ではなく 0 です。	使用不可	使用不可	使用不可	使用不可	ODBC の数値サポートを有効にするかどうか。
Object name end quote	"	- ODBC では、ユーザ入力ではなく DBMS からクエリされた値を使用します。	]	]	-	使用不可	SQL 文を作成する場合、終了文字が引用符として使用されます。
Object name start quote	"	"" ODBC では、ユーザ入力ではなく DBMS からクエリされた値を使用します。	[	[	-	使用不可	SQL 文を作成する場合、開始文字が引用符として使用されます。
PAD_BLANKS	使用不可	使用不可	0	0	使用不可	使用不可	スペース文字を使用して、一定のカラム幅を維持します。
Page size	使用不可	使用不可	使用不可	使用不可	使用不可	4096	512 以上で 32768 以下の、ページごとのバイト数 (2 の累乗である必要があります)。
Quote character	"	使用不可	使用不可	使用不可	使用不可	使用不可	QUOTE_START および QUOTE_END と同じです。
Quote object names	0 (使用しない)	0	0	0	0	0	1 に設定すると、QUOTE_START および QUOTE_END で指定された文字が、生成された SQL 文の識別子を囲むために使用されます。

DB オプション	インタフェースとデフォルト値						説明
	Oracle	ODBC/DB2	Sybase 15	Sybase	OLEDB	SQLite Persistent	
Reject log column delimiter	tab	tab	tab	tab	-	tab	拒否ログでカラム・デリミタとして使用されます。
Short column names	使用不可	使用不可	使用不可	使用不可	使用不可	0	フラグを false に設定すると、カラム名は完全に修飾されます。それ以外の場合は、カラム名によってのみ参照されます。
Show all tables	使用不可	使用不可	0 (使用しない)	0 (使用しない)	使用不可	使用不可	システム・テーブルとユーザ・テーブルを表示します。
Show error location	1	1	1	1	1	1	1 に設定すると、エラー・ロケーションを表示します。データベース・エラーには、結果セット内のレコードの位置が含まれます。
SHOW_ ERROR_ LOCATION_ ABSOLUTE_ ROWS	1	1	1	1	1	1	現在の結果セットではなく、結果セットの絶対的な先頭からのエラー・ロケーションを表示します。
Synchronous	使用不可	使用不可	使用不可	使用不可	使用不可	0	Full =2 を選択すると、SQLite が続行する前に、データがディスクに書き込まれます。 Normal=1 を選択すると、重要な状況では書き込みが一時停止されますが、Full ほど頻繁ではありません。 Off = 0 を選択すると、データがオペレーティング・システムに渡され、SQLite がすぐに続行します。
Temp store	使用不可	使用不可	使用不可	使用不可	使用不可	2	1 に設定すると、テンポラリ・データベースのロケーションはファイルです。 2 に設定すると、テンポラリ・データベースのロケーションはメモリです。

DB オプション	インタフェースとデフォルト値						説明
	Oracle	ODBC/DB2	Sybase 15	Sybase	OLEDB	SQLite Persistent	
Treat numeric value as character	使用不可	1 データベースが IQ であるか、ドライバ名に 'SYSYBNT' または 'LIBDB2.A' が含まれている場合、 ODBC ではユーザ入力ではなく 1 を使用します。	使用不可	使用不可	使用不可	使用不可	数値データを文字列に強制的に変換します。
Truncate reject log	1	1	1	1	1	1	データベース接続時にログがトランケートされます。値を 0 に設定すると、既存のログ・ファイルにデータが追加されます。
Use DELETE instead of TRUNCATE	0	0	0	0	0	0	1 に設定すると、TRUNCATE ではなく DELETE 文が使用されます。
Use system views	True	使用不可	使用不可	使用不可	使用不可	使用不可	DBA システム・テーブルを使用して、ユーザごとのメタデータではなく、メタデータを表示します。
Validate result column binding	使用不可	使用不可	使用不可	使用不可	1	使用不可	1 に設定すると、データベースからデータを読み込むときに、結果カラム・マッピングのバインドが検証されます。
Write empty dates as NULL	0	0	0 (使用しない)	0 (使用しない)	使用不可	使用不可	1 に設定すると、空の日付の値は強制的に NULL になります。
Write error code to reject log	1	1	1	1	1	1	エラー・コードを拒否ログに追加します。
Write error text to reject log	1	1	1	1	1	1	エラー・テキストを拒否ログに追加します。
Write header to reject log	0	0	0	0	0	0	拒否ログにヘッダ行を含めるかどうかを指定します。

DB オプション	インタフェースとデフォルト値						説明
	Oracle	ODBC/DB2	Sybase 15	Sybase	OLEDB	SQLite Persistent	
Write rejected records to file	-	-	-	-	-	-	拒否ログのファイル・パスを指定します。 このオプションは、ロード時にデータベースで拒否されたレコードのログを取るために使用されます。

注意：

- ハイフン (-) は、データベース・オプションにデフォルト値がないことを示します。適切な値を入力できます。
- 「使用不可」は、基になるインタフェースに対してデータベース・オプションが提供されていないことを示します。
- 「使用しない」は、データベース・オプションが表示されていても、基になるインタフェースで使用されないことを示します。

データベースとインタフェースのサポート

以下の表に、各データベース・コンポーネントのタイプで使用できるインタフェースとデータベースを示します。

表 B-1：データベースとインタフェースのサポート・マトリックス

インタフェース	DB Data Provider および DB Lookup	DB Data Sink	DB Bulk Load Sybase IQ	DB Staging	IQ Loader File via Load Table	IQ Loader DB via Insert Location
Sybase	ASE ASA IQ	IQ	IQ	ASE ASA IQ	IQ	IQ
ODBC	ODBC を介してアクセスできるすべてのデータ・ソース	IQ MS-Access	IQ	MS-Access IQ ASA ASE	IQ	サポート対象外
OLE DB	IQ SQL-Server	IQ	IQ	IQ	サポート対象外	サポート対象外
Oracle	OCI (Oracle Call Interface) によりアクセスできる すべての Oracle データベース・システム	サポート対象外	サポート対象外	サポート対象外	サポート対象外	サポート対象外

インタフェース	DB Data Provider および DB Lookup	DB Data Sink	DB Bulk Load Sybase IQ	DB Staging	IQ Loader File via Load Table	IQ Loader DB via Insert Location
DB2	IBM DB/2 クライアント・インタフェースによりアクセスできるすべての DB2 データベース・システム	サポート対象外	サポート対象外	サポート対象外	サポート対象外	サポート対象外
SQLite Persistent	すべての SQLite データベース・ファイル	サポート対象外	サポート対象外	サポート対象外	サポート対象外	サポート対象外

Sybase ETL 環境は、徹底的なテスト、評価と検証により、サポートされているデータベース・システムの多くのインタフェース・ドライバに準拠していることが確認されています。

ドライバの互換性がないことが原因で予期しない結果が発生していることが疑われる場合は、サポートされているバージョンのインタフェース・ドライバをインストールしてみてください。Sybase ETL 4.8 でサポートされているすべてのインタフェース・ドライバのバージョンのリストについては、『Sybase ETL 4.8 Release Bulletin』の「インタフェースのサポート」を参照してください。

SQLite Persistent インタフェースの使用

Sybase ETL テクノロジには、テンポラリ・データの格納およびステージングに使用される組み込み型の汎用リレーショナル・データベースが含まれています。これは、非常に高速で広範に使用されており、SQL-92 にほぼ準拠したデータベースである SQLite をベースとしています。SQLite は、自己完結型で埋め込み可能な、設定不要の SQL データベース・エンジンを実装する小さな C ライブラリです。

注意 SQLite はテスト環境でのみ使用してください。運用環境では使用しないでください。

SQLite データベースへの接続

SQLite データベースは、*.db* 拡張子が付いた単一のファイルとして表されます。データベース・ファイルには、任意の数のテーブルを含めることができます。

❖ **SQLite データベース・ファイルの作成または接続**

1 [Properties] ウィンドウの [Interface] メニューから [SQLite Persistent] を選択します。

2 SQLite データベース・ファイルのホスト名を指定します。

- 新しい SQLite データベース・ファイルを作成するには、[Host Name] フィールドで名前を指定します。*.db* 拡張子は含めないでください。*.db* 拡張子が付いた新しい SQLite データベース・ファイルは、デフォルト・ロケーションに自動的に作成されます。デフォルト・ロケーションは、インストール・フォルダの下にある *database* ディレクトリです。

  デフォルト以外のディレクトリに SQLite データベース・ファイルを作成するには、[Host Name] フィールドで *.db* 拡張子を付けたフル・パスを指定します。

- デフォルトのディレクトリにある既存の SQLite データベース・ファイルに接続する場合は、[Host Name] メニューからファイル名を選択します。*.db* 拡張子は入力しないでください。デフォルト以外のディレクトリにある SQLite データベース・ファイルに接続する場合は、[Host Name] フィールドで *.db* 拡張子を付けたフル・パスを指定します。

例 － 新しい SQLite データベース・ファイル *mySQLite.db* を作成する場合、または既存の *mySQLite.db* データベース・ファイルに接続する場合は、次のように設定します。

- インタフェース：SQLite Persistent
- ホスト名：mySQLite

作成されたデータベース・ファイルは、*mySQLite.db* という名前になります。

SQLite テーブルの作成

次のいずれかの方法で、SQLite テーブルを作成できます。

- Staging コンポーネントを右クリックし、[Create Staging Table from Input] または [Create Staging Table from port] を選択します。
- いずれかの Data Sink コンポーネントを右クリックし、[Add Destination Table from Input] または [Add Destination Table from Port] を選択します。

SQLite データベースからのデータの抽出

DB コンポーネントで SQLite データベース・ファイルの適切な接続パラメータを指定します。

SQLite でサポートされている SQL コマンドは、データベースに接続されているコンポーネントの前処理または後処理 SQL プロパティで使用できます。

[Tools] メニューの [Content Explorer] を使用すると、プロジェクトのコンポーネントに接続されている SQLite データベースのオブジェクトを操作したりブラウズしたりできます。また、*sqlite.org* で提供されているクライアント・アプリケーションを使用して SQLite データベース・ファイルに接続することもできます。

注意　外部のクライアント・アプリケーションを使用して SQLite データベース・ファイルに接続するには、SQLite のロック方式を理解しておく必要があります。

Oracle インタフェースの使用

表 B-2 は、Sybase ETL のデータ型から Oracle のデータ型へのクラス・レベル変換を示しています。

表 B-2：Oracle インタフェースから Sybase ETL へのデータ型マッピング

ETL のデータ型	Oracle のデータ型	サイズ／精度	最小の位取り	最大の位取り
binary	BLOB	2147483647		
binary	BFILE	2147483647		
binary	RAW	2000	0	0
string	CLOB	2147483647	0	0
string	CHAR	2000	0	0
float	DECIMAL	38	0	0
integer	NUMBER	38	0	0
float	DOUBLE PRECISION	15	0	38
datetime	DATE	19	0	0
datetime	TIMESTAMP	28	0	9
string	VARCHAR2	4000	0	0
unicode	NCHAR	1000	0	0
unicode	NVARCHAR2	2000	0	0
unicode	NCLOB	2147483647	0	0

付 録　C

緩やかに変化する次元に対するETL の使用

この付録では、緩やかに変化する次元 (SCD) の概要について説明します。いくつかの一般的な SCD シナリオを示し、ETL のプロジェクトおよびジョブを使用してこれらのシナリオを実装する方法について説明します。

概要

緩やかに変化する次元は、一般的なデータ・ウェアハウジング・シナリオです。SCD では、3 種類の異なる方法を使用して、データ・ウェアハウスの次元テーブルのカラムに対する変更を処理します。

タイプ 1 の SCD

タイプ 1 の SCD では、新しいデータによって既存のデータが上書きされます。既存のデータは失われ、変更履歴の追跡は行われません。タイプ 1 の SCD は、管理が簡単ですが、変更履歴の追跡が不要な場合にのみ役立ちます。

例：製品情報を保持するテーブルを考察します。

Key	Name	Price
1	Notebook	1200

Notebook の価格が 1500 に上昇します。更新されたテーブルでは、現在のレコードが上書きされます。

Key	Name	Price
1	Notebook	1500

タイプ 2 の SCD

タイプ 2 の SCD では、値の完全な履歴が維持されます。新しいデータが古いデータと異なる場合は、新しいデータ値で追加の次元レコードが作成され、現在のレコードになります。各レコードには、そのレコードがアクティブであった期間を示すための発効日と有効期限が含まれています。テーブルの次元データの完全な履歴を維持するには、タイプ 2 の SCD を使用します。

例：「ケース・スタディ・シナリオ」(318 ページ) を参照。

タイプ 3 の SCD

タイプ 3 の SCD では、別のカラムで変更が追跡されます。選択した属性の以前の値と現在の値は、1 つのバージョンの次元レコードに格納されます。タイプ 3 の SCD は、変更回数が限られている場合に、変更履歴を追跡するために使用します。

例：製品情報を保持するテーブルがあります。

Key	Name	Price
1	Notebook	1200

2008 年 7 月 15 日に、Notebook の価格が 1500 に上昇します。タイプ 3 の SCD に対応するために、Current Price と Effective Date という新しいカラムが追加されます。

Key	Name	Original Price	Current Price	Effective Date
1	Notebook	1200	1500	2008-07-15

注意 次元テーブルの構造の変更は、非常に重要な変更に限って行う必要があるため、タイプ 3 はあまり使用されません。

ケース・スタディ・シナリオ

この項では、タイプ 2 の SCD のケース・スタディ・シナリオを示し、Sybase ETL で変換プロジェクトを作成してこのシナリオを実装する方法について説明します。

ケースの説明

次の 2 つのテーブルがあります。

- 運用データベースまたはソース・データベース内の PRODUCT。
- データ・ウェアハウスまたはターゲット・データベース内の PRODUCT_PRICE。このテーブルでは、ある期間にわたって、次のようなソース・テーブル (PRODUCT) 内の製品の変更を追跡します。

- 既存製品の価格変更
- 新しく追加された製品
- 削除された製品

PRODUCT テーブルのデータベース・テーブル・スキーマは次のとおりです。

カラム	説明
Key	製品固有の ID
Name	製品の名前
Price	製品の価格

PRODUCT_PRICE テーブルのデータベース・テーブル・スキーマは次のとおりです。

カラム	説明
Key	ソース・テーブル (PRODUCT) のソース・キー識別子
Name	製品の名前
Price	製品の価格
Valid From	新しいレコードが挿入された日付
Valid To	レコードの有効期間の終了日。同じソース・キーを持つ新しいレコードが PRODUCT_PRICE テーブルに挿入されると、レコードは無効になります。

規則

PRODUCT テーブルから PRODUCT_PRICE テーブルに次元を転送するための規則は次のとおりです。

1 PRODUCT_PRICE テーブルにレコードが存在しない場合は、PRODUCT テーブルと同じカラム値でレコードを作成します。Valid From 日付をレコード挿入日に設定し、Valid To 日付を 9999-12-31 に設定します。

2 PRODUCT_PRICE テーブルにレコードが存在しても、変更されていない場合は、新しいレコードの挿入や既存のレコードの更新は行いません。

3 PRODUCT_PRICE テーブルにレコードが存在し、レコードの価格が変更されている場合は、次のようにします。

- 古い価格のレコードの Valid To 日付を、前日に設定します。
- 新しいレコードを作成します。Valid From 日付をレコード挿入日に設定し、Valid To 日付を 9999-12-31 に設定します。

4 PRODUCT_PRICE テーブルにレコードが存在しても、PRODUCT テーブルから削除されている場合は、PRODUCT_PRICE テーブルの製品の Valid To 日付を前日に設定します。

注意 次元の変更履歴は、これらの規則に基づいて、PRODUCT_PRICE テーブルに維持されます。

使用例

2008 年 1 月 1 日の最初のロード後、PRODUCT テーブルは次のように表示されます。

Key	Name	Price
1	Notebook	1200
2	Monitor	1000
3	Mouse	500

ETL プロセスを初めて実行した後、PRODUCT_PRICE テーブルは次のように表示されます。

Key	Name	Price	Valid From	Valid To
1	Notebook	1200	2008-01-01	9999-12-31
2	Monitor	1000	2008-01-01	9999-12-31
3	Mouse	500	2008-01-01	9999-12-31

2008 年 1 月 15 日に Monitor の価格が変更され、PRODUCT テーブルが更新されます。

Key	Name	Price
1	Notebook	1200
2	Monitor	1400
3	Mouse	500

ETL プロセスを再度実行した後、PRODUCT_PRICE テーブルは次のように表示されます。

Key	Name	Price	Valid From	Valid To
1	Notebook	1200	2008-01-01	9999-12-31
2	Monitor	1000	2008-01-01	2008-01-14
3	Mouse	500	2008-01-01	9999-12-31
2	Monitor	1400	2008-01-15	9999-12-31

2008 年 1 月 22 日に Hard Disk という新しい製品が追加され、PRODUCT テーブルが再度更新されます。

Key	Name	Price
1	Notebook	1200
2	Monitor	1400
3	Mouse	500
4	Hard Disk	1000

ETL プロセスを再度実行した後、PRODUCT_PRICE テーブルは次のように表示されます。

Key	Name	Price	Valid From	Valid To
1	Notebook	1200	2008-01-01	9999-12-31
2	Monitor	1000	2008-01-01	2008-01-14
3	Mouse	500	2008-01-01	9999-12-31
2	Monitor	1400	2008-01-15	9999-12-31
4	Hard Disk	1000	2008-01-22	9999-12-31

2008 年 7 月 28 日に PRODUCT テーブル が再度更新されます。

Mouse という製品がテーブルから削除されます。

Key	Name	Price
1	Notebook	1200
2	Monitor	1400
4	Hard Disk	1000

ETL プロセスを再度実行した後、PRODUCT_PRICE テーブルは次のように表示されます。

Key	Name	Price	Valid From	Valid To
1	Notebook	1200	2008-01-01	9999-12-31
2	Monitor	1000	2008-01-01	2008-01-14
3	Mouse	500	2008-01-01	2008-07-27
2	Monitor	1400	2008-01-15	9999-12-31
4	Hard Disk	1000	2008-01-22	9999-12-31

SCD のための ETL プロジェクトの設定

この項では、プロジェクトとジョブを使用してタイプ 2 の SCD を実行するための ETL の概念について説明します。製品に付属のデモ・リポジトリには、次のようなタイプ 2 の SCD の使用ケースに関連するさまざまな ETL 変換オブジェクトが含まれています。

プロジェクト

- Demo Product Price SCD – Initial Load

  このプロジェクトは、SCD - Update プロジェクトおよびジョブ用にデモ環境を初期化または再初期化します。

  注意 このプロジェクトは使用ケースの実装に含まれません。運用環境では、Update New and Modified プロジェクトを最初に実行すると、すべてのソース・レコードが新しいレコードとして処理される空のターゲット・テーブルで最初のロードが実行されます。デモ環境は 2 つの異なるテーブルを使用してソース・データの変化をシミュレートするので、このプロジェクトを実行してから他の 2 つの更新プロジェクトまたは更新ジョブを実行することによって、元のデータを常にターゲット・テーブルにリストアする必要があります。

- Demo Product Price SCD – Update New and Modified

  このプロジェクトは、ソース・データベースの製品の変更または追加を反映するように、ターゲット・データベースの次元テーブルを毎日更新します。「ケース・スタディ・シナリオ」(318 ページ) の規則 1 ～ 3 を参照してください。

  完全な更新を実行するには、Demo Product Price SCD- Update Deleted プロジェクトも実行します。

- Demo Product Price SCD – Update Deleted

  このプロジェクトは、ソース・データベースの製品の削除を反映するように、ターゲット・データベースの次元テーブルを毎日更新します。「ケース・スタディ・シナリオ」(318 ページ) の規則 4 を参照してください。

  完全な更新を実行するには、Demo Product Price SCD- Update New and Modified プロジェクトも実行します。

ジョブ

- Demo Product Price SCD – Daily Update

  このジョブは、Demo Product Price SCD - Update New and Modified および Demo Product Price SCD - Update Deleted の両プロジェクトを実行し、ターゲット次元テーブルの完全な更新を行うために 1 つの変換オブジェクトを作成します。このジョブを実行する前に、Demo Product Price SCD - Initial Load プロジェクトを実行してください。

ターゲット次元テーブルについて

ターゲット・レコードの識別

ターゲット次元テーブルには、同じソース・キーの複数のレコードがあります。過去のバージョンと現在のバージョンのレコードを識別するために、ターゲット次元テーブルでは、ソース・キーと発効日または有効期限の属性を含む複合キーを使用します。ETL のデモ・プロジェクトでは、キーの一部として Valid From 日付属性が使用されています。

現在のターゲット・レコード

ソース・テーブル内の各レコードは、ターゲット・テーブル内の 1 つの現在のレコードによって表されます。ターゲット次元テーブルをチェックするときは、現在のレコードのみが SCD に関連します。ETL のデモ・プロジェクトで、現在のレコードは Valid To の日付が '9999-12-31' になっています。

ソースの変更の検出

この項では、新しいソース・レコード、変更されたソース・レコード、ソースから削除されたレコードなど、ソース・テーブルの変更を検出する方法について個別に説明します。ここで説明する方法を組み合わせて、複数の種類のデータ変更を一度に検出することができます。

ケース・スタディ・シナリオでは、ソース・データベースに変更ログ情報が含まれないため、ソースとターゲットの内容を比較して変更を検出してください。多くの場合、ソース・オブジェクトとターゲット・オブジェクトは同じデータベースに格納されていないため、異機種間ジョインを実行する必要があります。

新しいソース・レコードの検出

ターゲット次元テーブルの最終更新後にソースに追加されたレコードには、対応する現在のレコードがターゲット次元テーブルに存在しません。

1 すべてのソース・レコードは、適切な Data Provider コンポーネントを使用して読み込まれます。Data Provider コンポーネントのリストについては、「Source コンポーネント」(107 ページ) を参照してください。

注意 検出に必要なのはキー属性のみですが、ターゲット次元テーブルに転送されたすべての属性が読み込まれます。

2 ソース・キー属性に基づいて、ターゲット次元に各ソース・レコードに対応する現在のレコードが存在するかどうかがチェックされます。

a 適切な Lookup コンポーネントを選択します。Lookup コンポーネントのリストについては、「Lookup コンポーネント」(147 ページ) を参照してください。

転送するデータに対して計算を行う必要がある場合は、Data Calculator コンポーネントの検索機能を使用することもできます。詳細については、「Data Calculator JavaScript」(130 ページ) を参照してください。

b ターゲットから検索データを選択します。これはレコードの存在を単にチェックするものであるため、すべての現在のターゲット・レコードからは元のソース・キーのみが必要になります。ただし、ETL の検索では必ずキーの値が返されるため、適切な戻り値も選択する必要があります。

c 検索結果を入力するための属性をポート構造に追加します。検索結果により、ソース・レコードが新しく追加されたか、または既に存在していたかが判別されます。この属性はデータの状態を示しており、これによって変換プロセスの次の段階でデータをフィルタできます。詳細については、「ポート構造の変更」(102 ページ) を参照してください。新しい属性は、Lookup コンポーネントの値属性または Data Calculator の出力属性として選択されます。

d　適切な検索デフォルト値を設定します。デフォルト値は、存在しないキーを検索したときに返されます。検索値がレコードの存在を正しく示すようにするには、既存のキーに対するすべての検索値とは異なる定数にデフォルト値を設定します。

例：

ソース・データ － PRODUCT から Key、Name、Price を選択します。

検索データ － PRODUCT_PRICE から Key、'1' を選択します。ここで、Valid_To は '9999-12-31' です。

値属性 － Exists (整数)

デフォルト値 － 0

この検索を実行すると、Key、Name、Price、Exists 属性を持つレコードが得られます。Exists は、ターゲットに存在するすべてのレコードに対して 1 (検索値) になり、存在しないレコードに対して 0 (デフォルト値) になります。

変更されたソース・レコードの検出

ターゲット次元テーブルの最終更新後にソースで変更されたレコードには、対応する現在のレコードがターゲット次元テーブルに存在しますが、関連する値は変更されています。

1　すべてのソース・レコードは、適切な Data Provider コンポーネントを使用して読み込まれます。Data Provider コンポーネントのリストについては、「Source コンポーネント」(107 ページ) を参照してください。

注意　検出に必要なのはキー属性のみですが、ターゲット次元テーブルに転送されたすべての属性が読み込まれます。

2　ソース・キー属性に基づいて、ターゲット次元テーブルに各ソース・レコードに対応する現在のレコードが存在するかどうかがチェックされ、値が比較されます。

a　適切な Lookup コンポーネントを選択します。

この場合は、1 つのキー属性に対して複数の値を検索し、比較を行う必要があるため、Data Calculator コンポーネントを使用します。詳細については、「Data Calculator JavaScript」(130 ページ) を参照してください。

b ターゲットから検索データを選択します。キー属性と、すべての現在のターゲット・レコードと比較するすべての値が、適切な Data Provider コンポーネントを使用して読み込まれます。Data Provider コンポーネントのリストについては、「Source コンポーネント」(107 ページ) を参照してください。

c 追加のターゲット・キー属性を入力するための属性をポート構造に追加します。検索結果により、ソース・レコードが変更されているか、新しいか、変更がないかが判別されます。この属性はデータの状態を示しており、これによって変換プロセスの次の段階でデータをフィルタできます。ターゲットの更新操作には、現在のレコードを一意に識別する必要があるため、ターゲット・キーの日付部分も入力する必要があります。詳細については、「ポート構造の変更」(102 ページ) を参照してください。

d 適切な検索デフォルト値を設定します。デフォルト値は、存在しないキーを検索したときに返されます。検索値がレコードの存在を正しく示すようにするには、既存のキーに対するすべての検索値とは異なる定数にデフォルト値を設定します。

e 必要なすべてのターゲット値を検索します。最初の検索では、新しいターゲット・キー属性が Data Calculator の出力属性として使用され、これにより存在が示されます。比較する値が、テンポラリ変数に読み込まれます。

f ソースとターゲットの属性値を比較します。既存のレコードの値の比較に基づいて、ターゲット・キー属性が再計算されます。

例：

ソース・データ － PRODUCT から Key、Name、Price を選択します。

検索データ － PRODUCT_PRICE から Key、Valid_From、Price を選択します。ここで、Valid_To は '9999-12-31' です

最初に、ターゲットから発効日を読み込んで存在をチェックします。

出力属性 － Valid_From

デフォルト値 － 0

テンポラリ変数に現在のターゲットの価格を読み込みます。

出力属性 － Tmp_Price

デフォルト値 － 0

現在のターゲット・レコードが存在する場合 (Valid_From が 0 でない場合)、Price と Tmp_Price を比較します。Price が変更されていない場合は、Valid_From を 0 に再計算します。

これらの検索と計算を実行すると、Key、Name、Price、および Valid_From 属性を持つレコードが得られます。Valid_From には、更新対象のターゲット・レコードの発効日か、新しいレコードと変更のないレコードを意味する 0 が格納されます。

削除されたソース・レコードの検出

ターゲット次元テーブルの最終更新後にソースから削除されたレコードには、対応する現在のレコードがターゲット次元テーブルに引き続き存在します。

1　ターゲット次元テーブルのすべての現在のレコードのキー属性が、適切な Data Provider コンポーネントを使用して読み込まれます。Data Provider コンポーネントのリストについては、「Source コンポーネント」(107 ページ) を参照してください。

2　ソース・キー属性に基づいて、ソースに現在の各ターゲット・レコードに対応するレコードが存在するかどうかがチェックされます。

a　適切な Lookup コンポーネントを選択します。Lookup コンポーネントのリストについては、「Lookup コンポーネント」(147 ページ) を参照してください。

ソース・データがデータベースに存在しない場合、Data Calculator コンポーネントの検索機能を使用します。詳細については、「Data Calculator JavaScript」(130 ページ) を参照してください。

b　ソースから検索データを選択します。これはレコードの存在を単にチェックするものであるため、すべてのソース・レコードからはソース・キーのみが必要になります。ただし、ETL の検索では必ずキーの値が返されるため、適切な戻り値も選択する必要があります。

c 検索結果を入力するための属性をポート構造に追加します。検索結果により、ソース・レコードが削除されているか、または引き続き存在しているかが判別されます。この属性はデータの状態を示しており、これによって変換プロセスの次の段階でデータをフィルタできます。新しい属性は、Lookup コンポーネントの値属性または Data Calculator 規則の出力属性として選択されます。詳細については、「ポート構造の変更」(102 ページ) を参照してください。

d 適切な検索デフォルト値を設定します。デフォルト値は、存在しないキーを検索したときに返されます。検索値がレコードの存在を正しく示すようにするには、既存のキーに対するすべての検索値とは異なる定数にデフォルト値を設定します。

例：

ターゲット・データ － PRODUCT_PRICE から Key、Valid_From を選択します。ここで、Valid_To は '9999-12-31' です。

検索データ － PRODUCT から Key、'0' を選択します。

値属性 － Removed (整数)

デフォルト値 － 1

この検索を実行すると、Key、Valid_From、Removed 属性を持つレコードが得られます。Removed は、既存のすべてのソース・レコードに対して 0 (検索値) になり、存在しないレコードに対して 1 (デフォルト値) になります。

その他の方法

- ソースが、挿入、更新、または削除のための昇順インジケータ (自動増分や変更日など) を提供するデータベースである場合は、DB Data Provider Index Load コンポーネントを使用して、最後のロード以降に変更されたレコードのみを読み込むことができます。「DB Data Provider Index Load」(111 ページ) を参照してください。
- ソースとターゲットの両方から同じデータベースに関連データをロードするには、Staging コンポーネントを使用します。その後、全外部ジョインを使用してステージからデータが抽出され、新しいレコード、変更されたレコード、および削除されたレコードが検出されます。詳細については、「DB Staging」(157 ページ) を参照してください。

レコードのフィルタ

データ・ストリームを複数の出力に分割する場合を除き、データ・ストリームからレコードを削除するには Data Splitter コンポーネントを使用します。Data Splitter コンポーネントの詳細については、「Data Splitter JavaScript」(140 ページ) を参照してください。

データ・ストリームからレコードを削除するには、削除するレコードと一致しないようにすべての出力ポートを定義します。1 つのデータ・ストリームを出力するには、デフォルト出力ポートの 1 つを削除して、1 つの出力ポートで Data Splitter を設定します。

ターゲット次元テーブルへのデータ入力

ターゲット属性への値の割り当て

DB Data Sink コンポーネントの挿入および更新オプションを使用すると、インバウンド・データ・ストリームに含まれないターゲット属性に値が割り当てられます。値には、定数または式の結果を指定できます。詳細については、「DB Data Sink Insert」(164 ページ) と「DB Data Sink Update」(169 ページ) を参照してください。

ETL のデモ・プロジェクトおよびジョブでは、挿入オプションで次のものを使用できます。

- 式 − Valid From 日付属性に対する '[uDate('now','localtime')]' (今日)。
- 定数 − Valid To 日付属性に対する '9999-12-31'

更新オプションでは、次の式を使用できます。
Valid To 日付属性に対する '[uDate('now','localtime','-1 day')]' (昨日)

部分更新の実行

DB Data Sink コンポーネントの更新オプションを使用すると、レコード全体を更新するのではなく、一部の属性を更新できます。属性を更新から除外するには、更新オプション・ウィンドウで選択を解除します。詳細については、「DB Data Sink Update」(169 ページ) を参照してください。

ETL のデモ・プロジェクトでは、Valid To 日付属性のみを更新することによって、古いレコードが除外されます。

付 録 D

ベスト・プラクティス

この付録では、Sybase ETL を使用した場合のベスト・プラクティスについて説明します。

ETL サーバを使用した場合のベスト・プラクティス

複数の ETL サーバ・セッションの起動を回避する

ETL Development の実行中に、コマンド・ラインから ETL サーバを起動すると、ETL Development は不安定になり、ETL Development でアクションを実行した場合、エラー・メッセージが表示されます。これは、コマンド・ラインから起動した ETL サーバ・セッションと ETL Development から起動した ETL サーバ・セッションの間で競合が発生するためです。

複数のサーバ・セッションが起動しないようにするには、[Preferences] ウィンドウで、[Engine] – [Start local engine during application startup] をオフにします。これによって、ETL Development を次回起動するときに、ETL サーバ・セッションが自動的に起動されなくなります。

コマンド・ライン実行のためのデフォルトのポート番号を入力する

デフォルトのポート番号 5124 を使用しようとしたが、これをコマンド・ラインに含めなかった場合、コマンド・ラインの実行に失敗します。この問題を回避するため、次のようにデフォルトのポート番号 5124 をコマンド・ラインに入力する必要があります。たとえば、次のように入力します。

```
GridNode -con --port 5124 --server localhost
-f "..¥¥testdata¥¥tpms¥¥tpms_TestB.xml"
```

ETL 4.2 リポジトリへのアクセス時にパスワードを再入力する

ETL 4.8 より前の ETL バージョンで作成されたリポジトリにアクセスするログイン時に、リポジトリ・ユーザ・パスワードを再入力する必要があります。

Query Designer でのクエリの入力時にカラム・エイリアスを使用する

Query Designer で、テーブルに関数を適用すると、Content Browser の出力カラム名が別のデータベースの別の名前になり、送信元カラム名とは関係なくなる場合があります。たとえば、次のようになります。

DBMS	関数	Query Designer のクエリ	送信元カラム名	出力カラム名
IBM DB2	CHAR	`select CHAR(col_1) from DATE_TBL`	COL_1	1
Adaptive Server	convert	`select convert(char(10), ID) from CUSTOMER_TBL`	ID	col_1
Sybase IQ	convert	`select convert(char(10), id) from alt_sales_orders`	id	id
Microsoft Access	SUM	`select SUM(SA_TOTAL) from SALES`	SA_TOTAL	Expr1000

カラムのエイリアスが出力カラム名として表示されるように、Query Designer にクエリを入力するときは、カラムのエイリアスを使用する必要があります。たとえば、カラムのエイリアス mycolumn を使用します。

```
select CHAR(col_1) as mycolumn from TEST_TBL
```

ETL コンポーネントを使用した場合のベスト・プラクティス

ワイド・テーブルのマイグレート

数百または数千のカラムを持つテーブルをマイグレートすると大量のメモリが消費されます。さまざまな数のカラムおよびローを持つワイド・テーブルをソース Sybase IQ データベースからターゲット Sybase IQ データベースにマイグレートするときに発生するエラーを回避するには、次の推奨事項に従ってください。

- Microsoft Access または SQLite Persistent リポジトリではなく SQL Anywhere リポジトリを使用する。
- 他のすべての値がシステム・デフォルトとして残っている状態で、SQL Anywhere リポジトリ用に 1GB のデータベース領域を割り当てる。
- 次に示す方法、コンポーネント、およびインタフェースを使用して、多数のカラムを持つテーブルをマイグレートする。

カラムの数	マイグレーション方法	使用するインタフェース
最大 3000 カラム	Insert Location コンポーネント	Sybase
最大 3000 カラム	Load Table コンポーネント	Sybase または ODBC
最大 3500 カラム	マイグレーション・ウィザード **注意** マイグレーション・ウィザードは、メモリ制限のため、または SQL Anywhere リポジトリを使用しない場合に、マイグレーション・プロジェクトの生成に失敗することがあります。	Sybase または ODBC
最大 10000 カラム	Load Table コンポーネント	ODBC

注意 3000 以上のカラムを持つテーブルをマイグレートする場合、多数のローの移動に伴うパフォーマンスの問題が発生する可能性があります。

32 を超える兄弟要素を持つ XML ファイルをインポートする

XML via SQL Data Provider コンポーネントを使用して 32 を超える兄弟要素をインポートするには、コンポーネントの [XML Options] プロパティで [Create Flat View] を 0 に設定します。Content Explorer を使用してサブクエリを手動で設定する必要があります。

ソース・テキスト・ファイルの最後のローを Sybase IQ にロードするには

最後のローが後続のロー・デリミタで終わっていない場合に、IQ Loader File via Load Table コンポーネントを使用すると、Sybase IQ では、ETL からソース・テキスト・ファイルの最後のローが受け入れられません。この問題を回避するには、行デリミタをソース・テキスト・ファイルの最後のローに追加します。たとえば、Windows で、CRLF として指定されたロー・デリミタを使用して、カーソルを最後のローの最後に置き、[Enter] キーを押して、ロー・デリミタを最後のローに追加します。

Adaptive Server Enterprise にバルク・コピーを設定する

DB Staging に Adaptive Server Enterprise を使用している場合は、最初に Adaptive Server データベースにバルク・コピーを設定する必要があります。Adaptive Server を設定しないと、プロジェクトの実行は正常に完了してもエラーが発生することがあります。

35 より少ない Data Calculator JavaScript コンポーネントを追加する

1 つのプロジェクトに 35 以上の Data Calculator JavaScript コンポーネントを追加しないでください。

Adaptive Server ODBC ドライバのテキスト・サイズを増加させる

Adaptive Server ODBC ドライバでは、Microsoft Windows のドライバの ODBC 設定で設定された値より大きい text または image データ値がトランケートされます。[コントロール パネル] の ODBC でテキスト・サイズの値を大きくするか、データベース接続のデータベース・オプションのパラメータの値を設定する必要があります。

❖ **[コントロール パネル] を使用したテキスト・サイズ値の増加**

1 [コントロール] - [管理ツール] - [データ ソース (ODBC)] を選択し、[ユーザー DSN] タブまたは [システム DSN] タブで Adaptive Server Enterprise のデータ・ソースを選択します。

2 [設定] を選択して、[ODBC Adaptive Server Enterprise Setup] ウィンドウを表示し、[詳細設定] を選択します。

3 [文字のサイズ] の値をデフォルトの 32 KB より大きな値に変更します。Adaptive Server ODBC ドライブでは、ここで設定した値より大きなデータ値がトランケートされます。

❖ **[Database Options] を使用したテキスト・サイズの増加**

1 データベース接続のプロパティ・ウィンドウで、[Database Options] フィールドの編集アイコンをダブルクリックして、[Enter Properties] ウィンドウを表示します。

2 [Extended Connect Options] フィールドに、“TEXTSIZE=*N*” と入力します。ここで、*N* は、設定するテキスト・サイズ値です。

Windows で Load Stage プロパティにファイル・パス名を使用する

DB Bulk Load Sybase IQ コンポーネントでは、[Use IQ Client Side Load] オプションを選択して、クライアント・マシンにあるファイルから IQ データベースへのデータのバルク・ロードを有効にした場合、[Load Stage] プロパティ・フィールドにパイプ名を入力する代わりにファイル・パス名を入力します。クライアント側ロードは、[Load Stage] のパイプ名ではサポートされません。

注意 クライアント側ロードは、ODBC をインタフェースとして選択した場合のみ有効です。

異なるプラットフォームでプロジェクトを実行する場合はソース・テキスト・ファイルのデリミタを変更しない

Windows で Text Data Provider コンポーネントを使用してプロジェクトを作成し、UNIX または Linux でそのプロジェクトを実行する場合は、ソース・テキスト・ファイルを UNIX または Linux フォーマットに変換しないでください。ソース・テキスト・ファイルで使用されるデリミタは、Text Data Provider コンポーネントで設計されたものと常に同じである必要があります。デリミタに一貫性がないと、実行エラーが発生します。

Windows 上での名前付きパイプのパーミッションの設定

DB Bulk Load Sybase IQ コンポーネントで、[Load Stage] プロパティ・フィールドにパイプ名を入力する場合は、エラーを回避するために、最初に次の設定を行う必要があります。

1 [コントロール パネル] - [管理ツール] - [ローカル セキュリティ ポリシー] - [ローカル ポリシー] - [セキュリティ オプション] を選択します。

2 [Policy] リストで [Network access: Named Pipes that can be accessed anonymously] をダブルクリックします。

3 名前付きパイプを既存のリストに追加します。たとえば、パイプ名が "pipe://mypipe" の場合は、リストに "mypipe" を追加します。[Apply] をクリックします。

4 [OK] をクリックします。

国際化を使用した場合のベスト・プラクティス

バイト順マーク付きソース・ファイルを正確に解析する

[Fixed by Bytes] プロパティを使用してファイルを解析する場合は、ソース・ファイルにバイト順マークを含めないでください。解析する前に、テキスト・エディタを使用してソース・ファイルからバイト順マークを削除します。

UTF-8 コード化をサポートするよう ETL を設定する

uSetEnv 関数に指定した引数には、マルチバイト文字または西欧言語以外の文字を含めることはできません。UTF-8 をサポートするように ETL を設定する必要があります。たとえば、

```
set LANG=zh.UTF-8
```

正しい文字セット・コードを選択して Unicode 文字を正しく表示する

“Text Data Provider” コンポーネントまたは “Text Data Sink” コンポーネントを使用して文字データをロードするときに、バイト順マーク (BOM) が含まれる Unicode ファイルの文字セット・コードに正しい「エンディアン」タイプを選択しない場合、文字が正しく表示されません。

Unicode 文字を正しく表示するために、コンポーネント設定ウィンドウの [Character Encoding] フィールドで、文字データの正しいエンディアン・タイプを使用する文字セット・コードを選択します。たとえば、次を選択します。

- UTF-16LE － ファイルの最初に BOM がある UTF-16LE でエンコードされたテキストファイルを処理する。ここで、BOM がファイルの最初にあるため、LE は「リトル・エンディアン」を指します。
- UTF-16BE － ファイルの最後に BOM がある UTF-16BE でエンコードされたテキスト・ファイルを処理する。ここで、BOM がファイルの最後にあるため、BE は「ビッグ・エンディアン」を指します。

索引

C

D

E

F

I

J

L

M

Q

R

S

T

X

あ

え

か

き

く

こ

さ

し

す

せ

そ

た

つ

て

と

な

ね

は

ひ

ふ

へ

ほ

ま

も

ゆ

り

ろ